Hewlett-Packard
Business Users Conference

I·N·T·E·R·E·X

ORLANDO
PROCEEDINGS

August 7-12
1988

VOLUME 2

RETURN TO:
HPL/RESEARCH LIBRARY
BUILDING #2L
P.O. BOX 10490
PALO ALTO, CA. 94303-0971
PHONE # 415-857-3092

QA76.8
H19 H16
1988
V.2

INTEREX

the International Association of
Hewlett-Packard Computer Users

Proceedings

of the
1988 Conference of
HP Business Computer Users
at
Orlando, Florida
August 7-12, 1988

VOLUME 2

William M. Lyons, Editor

HPL/RESEARCH LIBRARY
BUILDING 2L
P.O. BOX 10490
PALO ALTO, CA 94303-0971

An Evaluation of Data Base Performance Tools

Thomas Gosnell

Infocentre, Montreal, Canada

Overview

Fourth generation languages have become a common toolset, if not the toolset of choice, for most HP3000 development shops. The increase in use of fourth generation language and the large amounts of systems which have been built have resulted in the need for not only more functionality but greater performance.

During the early years of fourth generation language use, the need of data processing management was to increase programmer's productivity, release systems to the users faster and reduce the backlog of application systems. At the time, the driving force for leading fourth generation language vendors was to provide greater functionality and more productive means. Thus, succeeding in helping data processing departments deliver more application systems faster, more attention has been focused on optimising the runtime performance.

Leading fourth generation language vendors have in the past, and continue to address, the need for improved system performance by investing research and development resources in advanced alternate processing technologies. Two methodologies which have had success in improving Turbo Image../ Image throughput and performance are high speed indexing and fast serial read technology.

Turbo Image/3000, hereafter called just Image, has proven itself to be a reliable, flexible, efficient, and in many ways a very fast data base management system. Used by the majority of HP3000 application system users, Image access is a fundamental part of most fourth generation language's and can be arguably attributed to the success of many applications on the HP3000.

Image however has limitations and bottlenecks which limit fourth generation language processes from achieving maximum performance in both interactive and batch processing. Image's speed in interactive access can be excellent by using hashed master files to quickly retrieve data entries with complete key values. Yet, limitations exist preventing the rapid retrieval of data by secondary key value in the master sets, partial key retrieval in the master/detail sets, and relational conditions. Such constraints are addressed by building indexes on top of the existing Image structure to provide increased functionality and higher speed of retrieval.

Batch processing typically involves reading data sets serially, extracting the records meeting various selection criterias and subsequently processing the valid entries. This procedure is often restricted to the speed at which the records are accessed and read from the data base. There are parameters in both Image and MPE which may be changed to tune performance. These include buffspecs, blocksize, cache block size; all can make a significant improvement in performance or degradation but cannot overcome the overhead associated with Image serial read performance. However, fast serial read technology addresses the problem with impressive results. The present paper will review these complementary technologies in a generic fashion, providing the reader with some general guidelines for the best fit scenarios.

Capability sets and performance will differ between vendors of similar products. It is the purpose of this paper to illustrate the comparative advantage of the different technologies and not provide product comparisons or definitive statements on performance. As such not all listed capabilities will exist in similar products, nor will performance figures be the optimal available. All tests were executed on a HP3000/930 running MPE/XL A.01.20 in compatibility mode.

Indexing Systems

There exists many types of indexing techniques and technologies which increases the speed in which data may be accessed. Every viable algorithms possesses characteristics which assists in defining performance and functionality versus resources consumed . Hashing techniques, such as used by Image master files are arguably the fastest least consumptive methodologies available, but require that key items will hash evenly and will be suitable for retrieval by their full key value.

In the real world of data processing, many situations arise whereby user retrieval requirements dictates a data model which does not correspond to a proper fit to the Image data base management system technology. There are without questions, techniques and tricks which have been used to force fit Image with application systems. Such methods are not always efficient or easy to program using a fourth generation language.

An alternative data indexing technique to hashed master files is the binary tree concept, used with KSAM on the HP3000 and HPIMAGE on the 9000/8XX machines. B-Trees possess many characteristics which make them a very good alternative and an excellent complement to hashed master files.

B-Trees, unlike hashed techniques keep key values in sorted order with index blocks and pointers to quickly locate key items with partial as was well as full key values. Such characteristics allow B-Trees indexes to serve as foundation to full feature data indexing and enhanced data base management system functionality.

Implementation of the technology differs for enhanced Image/fourth generation language data base access, but typically they consist of various programs and intrinsics which support the complete development and runtime environment. Such routines allow the definition, installation, maintenance, programming and runtime support needed for external access.

Capabilities

Data entry retrieval for indexed data items are possible by:

- item value;
- partial item value;
- key word;
- partial key word;
- multiple item;
- multiple item, multiple values;
- relational conditions between items;
- multi set retrieval;
- field grouping

Implementations

Index data structures are built using vendor specific techniques, but typically:

- contain a B-Tree structure for partial value indexed item retrieval;
- build data structures within standard Image data sets;
- create an index root file to describe indexing characteristics;
- provide transparent and non transparent support from fourth generation languages;
- installation programs for configuring and reconfiguring indexing requirements;
- maintenance programs to rebuild indexes on new or old data bases;

Operational characteristics

- Automatic recognition of indexed data sets by fourth generation language processor.

- During inquiry or modification mode, indexed data items are prompted for, at which point any combination of the retrieval capabilities can be used.
- Entries are qualified from indexes and not retrieved from data sets during inquiry processing producing a very responsive interface.
- Qualified entries can be listed data using relational conditions, displayed and modified if required.
- Qualified entries can be passed to reports for printing or complicated outputs.
- Indexing cannot easily span data bases without the use of regular image keys.

Programming Effort

- Indexed data sets are automatically recognized and require no effort to maintain from a programming stand point.
- Advanced reporting on multiset displays typically have to be coded in the reporting language.
- Existing screen displays will typically benefit from retrieval capabilities with no modifications.

Performance Characteristics

-Retrieval from Data Base

- Qualifies entries very quickly; example: Master data set 8192 entries, 18013 capacity;

single item value	160 records	< 1 second
three values using "or"	370 records	< 1 second
single partial value	212 records	~ 1 second

- Multiple items from single data set;

single item and	160 records	< 1 second
second item	160 records	< 1 second
	TOTAL TIME	< 1.5 second

- Qualification and Retrieval

single item value	160 records	4.5 seconds (1.8 seconds)
multiple item value	370 records	10 seconds (4 seconds)

- Average Rate of Retrieval	37 records/ seconds	91 records/ CPU second

- Adding to Data Base

Indexed data items require significant amount of time to be placed into a data base. The number of index data sets which need to be updated for each new entry can be as little as one and as many as four for each indexed item! Therefore, the performance of adding to a data set can vary and be significant. Typically in on line entry with four or five items indexed, the time to add to the data base will be less than 1.5 seconds. This amount of time is very often acceptable for low volume of fairly static information. In high volume data entry applications, the time to add may be too great to sustain an acceptable level of performance. In general, each item indexed will have a cost similar if not slightly greater then adding a chained item in a detail data set.

In addition to the performance cost of adding records there exists a cost in disk space. The data base in which the performance data was extrated was originally contained within approximately 34,000 sectors. After a moderate amount of indexing, 4 items totalling 50 bytes in a master of 8930 records and one 82 byte field in a detail data set of 62,190 records the data base required an additional 7100 sectors of disk space or approximately 25% more. It should be noted however, if the application did not use indexing, it may not have been possible to implement with a fourth generation language or Image.

Using the Technology

Index systems offer functionality and performance which is very appealing. The ability "to find the blue needle in the hay stack" in under a second is very attractive. Care must be taken however, not to try and replace all Image search items with indexed data items or index indiscriminantly.

Image search item structure are efficient and provide excellent performance, where full key values are known or when the data base is being accessed programmatically.

Index data structures are less efficient in CPU and disk space usage, but provide a more flexible and improved responsive user interface. These characteristics indicate that using indexes should be viewed as a complement to Image search structures and not a replacement.

Good candidates for using index data items are master type files where data is relatively stable and textual information is present; two examples of such files are the customer master and part master.

In both of these data sets, numbers and/or codes are typically assigned to each record. These represent a unique occurrence of data and represent an item with textual information.

Without indexing an interactive user would normally be required to know the unique record identified, be it part number or customer number to access stored information. Using indexes, users may query the data base with more meaningful data items such as names, cities, or part descriptions and interactively retrieve the information required without knowledge of codes or numbers.

Poor example of using index data items would be high volume transactions data sets, such as an order detail or inventory transaction file. While their may very well be data that would benefit from indexing, performance of maintaining index data's structures, in high volume applications, may slow data base performance to unacceptable limits.

Reporting on data sets which have been indexed may benefit from using the indexes, but care must be taken as the maximum throughput of reading data with indexes is less than with other techniques. Maximum rates qualifications and retrieval are about 70 records/second. Other techniques, such as fast serial read can be processed at a rate of over 2000 records/second. Indexing however, does not require that the entire data set be read and can therefore offer superior performance for small subsets of data.

High Speed Serial Read

High speed serial read technology enhances data base access from fourth generation languages by increasing the rate at which data can be extracted. Using a set of decision rules, the language processor will invoke special data base access routines when possible to dramatically increase the speed at which data can be read from a master of detail data set.

Requiring no prior configuration or set up, high speed serial read is very easy to use and to benefit from. Maximum optimization of the technology requires that the user clearly comprehends both performance characteristics affecting the data as well as the data itself.

Implementation

- High speed data access intrinsincs use privileged mode to open data sets.
- Data access intrinsincs are typically contained within the system segmented library, thereby eliminating the need for privileged mode compatibility on the fourth generation language

processor.
- Language processor has knowledge of processing context and with built in decision rules can activate high speed routines if needed and available.
- Access intrinsics routines use multi record NoBuff IO.
- Access routines may be disabled by system manager or programmers.

Operational Characteristics

- Forward serial read only (DBGET mode 2);
- Data cannot be updated while being read;
- Verifies and respects Image security;
- Bypasses Image control blocks to reduce overhead and increase speed;
- Requires a minimum of 4,000 words of stack space.

Performance

- As no data modifications must be performed, there is no effect on adding data to the data base.
- Indirectly using fast read can reduce the need of search items, or indexed data items by providing alternate means to access data.
- Forward serial read is approximately 6 items faster than a regular serial read.

Set Type	Capacity	Entry Size	Reading Technique	Elapsed Time	CPU Time	Records/ Second	
Master	8930/18013	101	Image	29	20	307	
Master	8930/18013	101	Fast Read	42	3.9	2126	(6.7 times)
Detail	62190/91008	51	Image	126	124	493	
Detail	62190/91008	51	Fast Read	22	18.5	2826	(5.7 times)

Using the Technology

The raw speed of fast read technology is impressive and useful by itself, but to gain maximum benefit, some thought must be given to reading algorithms for the given structure a data base.

To maximise the gain of using fast read technology, data sets which require lengthy processing should be reviewed so as to decide whether serial read may be implemented. An example follows which illustrates three different approaches, with and without fast read to help comprehend the performance characteristics in reading a master and detail data set.

- Data base Layout (subset):

Set	Type	Capacity	Entries	Seconda -ries	Average chain
M-Product	Master	18013	8930	21	1.25
D-Description	Detail	91008	62190	-	7

The processing objective is to read each master record and its related detail. In the first two cases, for each master record a chained read is performed on the detail data set. In cases 3 and 4, the detail data set is read serially and for each record, a calculated read is performed on the masted data set. In tests 5 and 6, the detail data set is again read serially , but a calculated read is performed when the current master record does not have the same key value as the current detail.

Cases	Elapsed Time	CPU Time	% Case 1	% Case 2
1 Read master chain detail	186	178	100	105
2 Fast read master chain detail	177	174	95	100
3 Read detail-calculated read master	318	315	171	179
4 Fast read detail-calculated read master	225	220	121	127
5 Read detail-calculated read master-break	213	211	115	120
6 Fast read detail-calculated read master-break	109	108	59	61

The best results illustrate the potential of using alternate processing methods. Case 6 in which the detail data set was read serially with fast read and the master data set read for each new key value resulted in a performance increase of 67% over case 2 where we read the master serially with fast read and we chained to the data set.

Their exists a particular data characteristics which had an effect on the results that is, the packing of the detail data set. This one fact resulted in the optimization of cases 5 and 6 by eliminating over 53,000 calculated reads. This same fact also greatly optimises cases 1 and 2, by producing the nearly optimal chain read environment.

While the ideal packing skewed the results, it should be kept in mind that all data bases have particular data characteristics. The knowledge of data particularities and of performance profiles of the various access techniques are mandatory for achieving maximum performance.

By retrieving the results, the following performances profiles can be extracted:

Fast serial read = 2100 records/second for master data set at 50% capacity
= 2800 records/second for detail set
Regular serial read = 307 records/second for master data set at 50% capacity
= 493 records/second for detail set
Calculated read = 320 records/second for master data set at 50% capacity
Chained read = 350 records/second for detail set
Logic process for conditional check = 1000 records/second

These results should not be taken to suggest that detail sets should be read first but rather that details can be read first and thus eliminate the need for a detail search path and still be faster.

The replacement of detail chains used for reporting purposes by run time access routines have the potential of not only increasing the speed of reading data but also decrease the overhead associated with adding records to the data base.

In summary, fast read routines will improve the rate at which data can be read from the data base. The passive use of this technology will provide small gains in performance. More conscious use however, will provide the analyst with the possibility of gaining significant run time reductions and the opportunity to decrease data base index structures.

Conclusion

Fourth generation language vendors are supplying alternative ways of addressing the need for greater input/output performance from Image data bases. Easy to use interfaces and, in some cases fully transparent interfaces, may use these techniques very easily. However, the numerous performance parameters and the greater number of processing options require that additional analysis be performed. Informed use will increase system throughput and provide more productive systems. Negligent use can result in no improvements and even possible performance degradation.

With the maturing of fourth generation language products, data processing system production is less time consuming, however, to fully exploit their capabilities requires very detailed analysis and excellent understanding of the development tools.

TITLE: A Rational Use of Micro Computer Resources

AUTHOR: Andre Courtemanche

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 0101

TITLE: Becoming Effective Tradespeople

AUTHOR: Winsome R. Stretch

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 0107

Telephone Support - Proven Ways to Maximize Your Investment

Doug Clement
Cognos Incorporated
3755 Riverside Drive
P.O. Box 9707
Ottawa, Ontario
Canada K1G 3Z4

Introduction

Software support contracts represent a significant annual expense for most organizations. In order to maximize this investment it is important that the support be of the highest calibre. Most maintenance agreements ensure customers that the product they have purchased will remain current within changing operating environments. In addition, these agreements ensure that customers' needs are met by introducing new features to products and by providing access to a support center that delivers regular problem resolution and addresses education and information requirements. But there are other methods of ensuring a sound return on the initial software investment. This paper outlines some of the proven ways for both the customer and supplier organization to achieve this goal.

Remote customer support, generally delivered by telephone, is the cornerstone of most support agreements. It provides an effective channel for immediate customer assistance when problems are encountered, and keeps the supplier organization up to date on user satisfaction and future requirements. Typically it is a high-volume service that offers technical information about known problems and solutions, guidance on correct product usage, and assistance in the analysis of problem situations. Although dependent on the nature of a specific product, the target audience for such a service is usually the technical user designated by the customer organization as the focal point for communication between the product users and the supplier.

In the course of dealing with thousands of customer organizations, many different opinions are expressed regarding the quality of support provided. Customer opinion varies widely, from complete satisfaction and therefore a perception of value received, to dissatisfaction with the quality or responsiveness of the support organization. This variance in satisfaction is experienced by customers who are using the same products, encountering similar difficulties and dealing with the same support unit for assistance.

The question that arises is why are a majority of callers pleased with the service while others are not? A combination of factors relating to both the service and customer organizations can account for the varying degree of satisfaction.

Service suppliers face an interesting set of challenges that, if not addressed, can impact short- or long-term service delivery. Some ot these challenges include:

- a higher than normal staff turnover due to the nature of "hotline support";
- rapid vertical and horizontal product-line expansion into new operating environments;
- increased product sophistication and technical complexity;
- difficulty in recruiting staff with the appropriate combination of technical and communication skills;
- increased customer requirements and expectations;
- continued pressure to control costs.

Support center managers are constantly exploring ways to address these issues and to deliver a consistently high-quality service.

Possible solutions to these problems constitute an interesting topic for discussion, but are not the thrust of this presentation. A focus on the means of obtaining the most from your support investment is not intended to minimize the challenges facing support organizations.

Factors within the customer organization that can contribute to a successful relationship with the service supplier include:

- the degree to which supplier contact is focussed;
- product knowledge and access to product information;
- the expectations of a remote support service within an overall support strategy;
- how change in the technical environment is managed.

Maximizing Your Investment

How can you maximize the benefits available from a remote support service? Following are some proven techniques:

- ### Establish the Correct Support Center Liaison

  Focussing contact with the service supplier through a small group within your organization will significantly improve the relationship between you and the vendor and will result in faster problem resolution.

  By channelling communication through a few individuals, a familiarity is developed between those who require assistance and those who provide it. Through repeated contact, product expertise is developed and centralized, increasing the ability of a customer's designated contact to solve problems immediately without further assistance. Knowledge of the correct approach to problem analysis and investigation is also developed, leading to a faster problem resolution.

The customer's in-house designated contact should possess the skills and knowledge needed to function effectively as a focal point for assistance and escalation. They must have strong communication skills, proven analytical and problem investigation capabilities and the judgement necessary to prioritize issues. Product expertise can be developed by taking advantage of vendor-supplied courses at both the introductory and advanced levels prior to regular and practical application of the product knowledge. Appoint alternate contacts so that if a change in assignment occurs, or your designated contact is unavailable, the support service can still be used.

Our experience at Cognos indicates that relationships established between our support staff and designated customer contacts result in an efficient and productive framework for service delivery. Another benefit of this focussed communication is a reduced demand on the support center staff, allowing them to devote more time and attention to each issue raised.

• Get to Know Your Service Supplier

The wide range of responses concerning the quality of support can be explained by differing expectations within the user community. Obviously if the expectation is low, it is much easier to satisfy. Suppliers maintain different goals and objectives for their service organizations. They may see it as a necessary service, but not one that constitutes a key component of their customer communications strategy. Others may have much more ambitious goals for their support vehicle.

You should know how the service department that you deal with fits into their corporate organization. What types of questions are they positioned to deal with? How do they view their role? How do they measure their success? When should they be consulted and what results can you expect?

If these questions are not answered in your support contract or through support publications, call the support management. Discuss the structure of their organization, the problem process and other general policies and procedures.

Understanding the role of your support center, their escalation procedures and their management structure can be very helpful when exceptional action is required to deal with a sudden crisis.

• Organize Reference Material

Software manufacturers dedicate considerable resources to supplying comprehensive product literature to resolve anticipated customer problems and questions. This reference library can be an invaluable tool in a variety of situations. At the Cognos North American TeleSupport Center, over 25% of the calls received are solved by simply referring customers to information previously provided. Placing a call therefore introduces an unnecessary delay into the resolution process. By organizing all related reference material and referring to it before placing a call, you can often resolve problems in a faster and more efficient manner.

A variety of information may be sent with a product release and, later, at regular intervals. Material may be provided in scheduled newsletters, documentation notes on your tape or diskette, manuals or other supplementary publications. Typically customers receive:

- manual updates;
- installation instructions;
- conversion guides;
- descriptions of new features;
- problem notifications;
- tips and techniques about product usage and performance;
- surveys and questionnaires.

It is recommended that you assign responsibility for maintaining a current inventory of all related product material to one individual. Although many people may receive product information, it is important that one person maintain and disseminate this information from a central source.

• Educate Technical Staff

A small investment in product education for your technical staff can result in tremendous productivity gains and faster problem resolution.

As technology advances, companies strive to make their products more powerful, more sophisticated and easier to use. Whether you are riding a bicycle, piloting a plane or using a sophisticated piece of software, you will do it better and require less assistance when you are properly trained. Education is particularly important for the individuals assigned as your designated contacts for support. They are the ones chosen to be your in-house experts. Their increased level of product knowledge will result in fewer, and more productive contacts with the remote service center.

The investment in training is often very small in relation to the cost of software and to the costs introduced by unnecessary project delays due to inadequately trained staff.

Many options are available to minimize on-going training costs, such as "train the trainer" programs, computer-based courses, or self-paced audio or video packages that can be purchased once and used as refreshers or to train new staff. Examine the options available and include the development of an on-going training program as part of your product orientation program. Remember to allow for enhanced levels of training and product updates at regular intervals. The advanced knowledge and skills gained from this product education will guarantee that your investment is realized to its full potential.

- Maintain a Current Environment

Support contracts typically provide for the periodic delivery of new software releases and new documentation. New versions address three major requirements:

- the need to change software to stay compatible within the operating environment. New operating systems, changes to hardware, or upgrades to other system software may force the software supplier to enhance a product.
- the need to add new features to encourage sales and customer acceptance by responding to market requirements.
- the need to resolve outstanding, high-priority problems identified in earlier releases.

When a new version is released you may not want to undertake the risks associated with an upgrade, particularly if you have just reached the point where everything is running smoothly. The temptation may be to leave things as they are.

From the supplier's perspective, their efforts are best directed towards supporting only the current version, following a reasonable length of time to upgrade. They will concentrate their training, problem resolution and support center resources exclusively on the latest release. A problem encountered in an obsolete version may be recognized as a problem only if it also exists in the most recent release. Resources allocated to resolving it will be focussed on the latest version only.

The decision not to upgrade to a new release can lead to a potentially dangerous situation. The only solution to a problem affecting your production environment may be the immediate installation of the most recent product offering. Production pressures may force an abbreviated upgrade schedule that does not allow sufficient time to assess the impact of product changes and new features. Alternatively, it may be necessary to consume extra resources by running a parallel environment for a period of time.

Scheduling regular upgrades of all new versions to occur within the allowed time avoids panic situations and ensures that the software supplier's efforts in supporting a product have direct benefits to you. The upgrade process can be streamlined by maintaining a comprehensive library of test cases to be used when required.

If an upgrade is simply not possible due to production pressures or other scheduled system changes, take an interim step of installing the new version in a test environment. This will leave your production environment intact. Prior to raising issues in older versions with the support center, test them thoroughly in both. If the problem exists in the current version, it will receive the proper attention when reported.

- Isolate Problem Situations

Proper isolation and investigation of software problems will result in faster, more complete responses from the support center.

The most difficult task facing remote support staff is translating a problem description communicated over the phone into a test case that substantiates the problem and can be used to verify possible solutions.

A tremendous amount of time can be spent in discussing the problem, dialling in to investigate, and simulating it in the support center's environment. The time invested in isolating and simplifying the problem area before contacting the support staff will be repaid many times over through faster problem resolution.

This benefit can be realized if specific steps are followed to effectively prepare prior to calling the support center. Service organizations can often provide a checklist to follow before you make contact. The list will usually request information such as:

- a description of the hardware and software environment, including all version numbers and machine model numbers;
- the exact text of system or application error messages;
- the solutions or workarounds that have been tried and the results obtained;
- the sequence of events that are required to reproduce the problem;
- any system or application changes introduced immediately prior to the occurrence of the problem;
- any patches or temporary fixes that have been applied;
- an accurate assessment of the problem's impact on the production environment;
- the information necessary to identify the customer organization.

Initially, these checklists may seem to be an administrative burden. They are, however, effective in ensuring that all the required information is available to the support specialist. The timeliness of a response can be influenced tremendously by the quality of information initially provided.

- Use the Best Communication Channel

Alternate forms of communication with a remote support center can dramatically improve the quality of information and the speed at which it is exchanged.

Communicating technical issues over the telephone can be very difficult. Establishing contact on the first call can sometimes be impossible. When contact is made, the process of detailing a lengthy description of an intermittent problem can exasperate both parties. The results can be less than satisfactory.

A recent trend within support organizations has been to provide alternate communication channels, better suited to this type of information exchange. Regular mail service is available for low-priority problems, or telex for brief exchanges. Facsimile transfer has a role, but is not universally available. With the advent of personal computers, dial-up bulletin boards and electronic mail services, have opened up a wide range of possibilities for fast and accurate communication.

There are advantages for both the customer and the support center when electronic channels are used. The service supplier can manage problems more efficiently by stacking them as they arrive. In this way the customer avoids waiting by the telephone for a response, and both benefit from a more precise exchange of information. Bulletin boards or broadcast messages to all subscribers can reduce calls by supplying information before it is needed, as in the case of a known software defect not yet encountered by all customers.

Cognos has been using an E-Mail network to communicate with branch offices around the world. Recently customers have been joining as well. The result has been a more orderly support envrionment and an improved, streamlined level of service.

If your supplier is accessible via such a service, use it on a trial basis. If this facility is not yet available, discuss a test implementation on one of the public E-Mail services. Increasingly, electronic mail services are interconnected and governed by international standards. It may not be necessary to subscribe to the same service as your supplier in order to communicate. By using electronic mail, you can experience a significant improvement in the quality of communication.

• Work with the Support Center Staff

Maintaining your involvement throughout the problem-solving process will improve both the quality and timeliness of the solution. Your skilled and knowledgeable staff together with the support specialist will form a strong problem-solving team.

Responding to a wide variety of questions and problems is an inexact process. Both parties hope that a straightforward question from a customer will lead immediately to a complete and accurate response from the support center. Unfortunately, the complexities of hardware and software often place those charged with resolving problems in the role of detectives, sorting through large amounts of information to find the key to the solution.

The process of arriving at the simplest problem definition may require repeated attempts at determining important information, comparing results in different environments, and trying several possible solutions. This iterative approach calls for an investment of time and energy by both the customer and the support staff.

To achieve a satisfactory resolution it is vital that the customer's specific knowledge, the resources of the support center, and the creativity of both focus on the problem. This process involves an investment of your time along with that of the support specialist in order to effect a satisfactory conclusion.

• Capitalize on Alternate Support Services

Remote support assistance usually represents just one of the many services available to customers. Properly applied, these service programs can provide specialized assistance in areas beyond the capabilities of remote support.

Recognizing this, suppliers develop a wide range of services as part of their overall support strategy. These services are complementary and designed to meet the needs of a diversified customer base. Specialized consulting throughout the application-life cycle, assistance with unusual education requirements, or product start-up programs are examples of services frequently offered.

A problem raised with the support center may require the specialized resources of one of the other service components. The supplier's marketing staff or service center management can advise on alternate services and how they can be productively applied to various problem areas. As part of a product implementation plan, consider the specialized assistance available and how it can be used to ensure continued success in the use of your purchased software.

• Provide Feedback

Customer input enables a support organization to build on its strengths and correct its weaknesses. The result is a tailoring of services to better address the needs of its customers.

One challenge in running a support organization is determining the extent to which customer requirements are successfully addressed. Internal measurements can be used to monitor such things as call volumes and response time, but these indicators do not accurately gauge customer satisfaction.

Measuring satisfaction requires regular and frequent feedback from callers and responses to mail or telephone surveys. Mail surveys typically have a low response rate. This represents a missed opportunity for customers to impact the quality of service being delivered. When a survey crosses your desk, or when your maintenance renewal contract arrives, take a few minutes to let your supplier know what you think of their service. At any convenient opportunity, make your views known. Over time, customer input can result in positive improvements to service offerings.

Summary

A remote technical support center provides a key service that, when used correctly, can help to protect the value of your product investment. Understanding the role of the support center within an overall support strategy, and taking the time to thoroughly prepare before placing a call will increase service benefits. Additionally, keeping the communication channels open will help the supplier to understand and respond to changing customer service requirements.

OPTIMIZING THE LOGICAL DATABASE DESIGN

DICK ONEL

DCE DATABASE CONSULTANTS EUROPE
PRINSENGRACHT 747-751
AMSTERDAM
HOLLAND

ABSTRACT

The logical database design of an IMAGE database consists of those aspects of the design that determine it's functionality namely it's structure, in terms of data sets, and the entry points to those data sets. When the logical database design has been enhanced with the physical parameters and reorganisation procedures which determine the physical organisation of the database, we speak of the physical database design.

The physical database design process is oriented at obtaining optimal performance. However, there are also many ways to optimize in the logical database design which result in better performing and longer lasting database applications. If these applications have unsatisfactory performance, or if they use more resources then necessary or available, it is, among others, the logical database design which must be re-evaluated. In order to do this, we must understand the transaction(s) causing the problems, know Turbo-Image, and have resources, such as disk space, or CPU cycles, to make design trade-offs with.

The logical database design is made on the basis of assumptions of how the data will be used, and should support all required accesses to the data. When the database is in production the real usage of the data should be re-evaluated. Based on these re-evaluations the access structures should be revised , i.e. remove not used or infrequent used paths and/or add new paths. The logical database model should theoretically have no redundant data. If this model shows performance problems we should optimize it. We can add redundant data in masters and/or details or add redundant data and access structures. We can split entries , distinguish between files and tables and between current and historical data.

These measures should make a stronger foundation for the future of our databases.

INTRODUCTION

The logical database design of an IMAGE database consists of those aspects of the design that determine it's functionality - namely it's structure, in terms of data sets, and the entry points to those data sets. When the logical database design has been enhanced with the physical parameters and reorganisation procedures which determine the physical organisation of the database, we speak of the physical database design. The physical database design process is oriented at obtaining optimal performance.

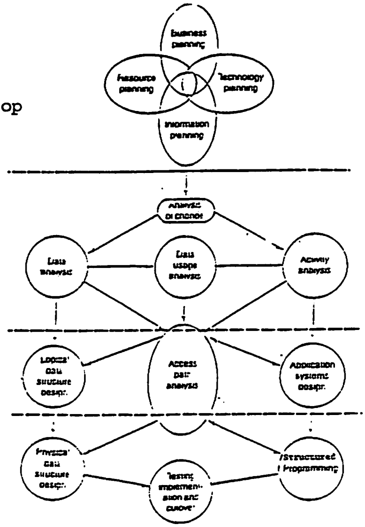
However, there is no point to optimizing the physical storage of the database if the logical database design does not match the requirements of the business. A poor logical database design is often the fundamental cause of performance problems in Turbo Image databases.

In order to produce good logical database designs, we need methods and techniques which will enable us to identify what the business requirements mean in terms of what data should be stored in the database, what the inherent logical structure of that data is, and how it will be accessed.

All design begins with analysis , even if prototyping techniques are used. The analysis should be carried out in a structured manner and produce documentation and diagrams that feeds into the design process.

The Systems Development Cycle, shown in figure 1, is a proven approach for developing database systems, and forms the basis of the techniques discussed in this paper.

Figure 1.
The Systems Develop
Cycle


business planning
resource planning
technology planning
information planning
Analysis of change
Data analysis
Data usage analysis
Activity analysis
Logical data structure design
Access path analysis
Application systems design
Physical data structure design
Structured Programming
Testing implementation and cutover

The right hand side deals with activities, functions or processes. The left hand side deals with data. Recent studies show that the data side of an application system is far more stable than the activity side. It is therefore imperative to have separate analysis and design processes for determining the data structure, and not just let the data structure "fall out" of the functional analysis and design.

The first part of the analysis phase is defining the application-system aims and the scope of the analysis; this is called the BUSINESS ANALYSIS or FEASIBILITY-STUDY.

The second step is the analysis of the data resources used and the analysis of the user's information handling processes, the DATA ANALYSIS and the ACTIVITY ANALYSIS. To cross-reference, and check the consistency of the Data Analysis and Activity Analysis, DATA USAGE ANALYSIS is performed.

At the next step, the LOGICAL DATA STRUCTURE DESIGN (or LOGICAL DATABASE DESIGN) is performed, using the results of the ACCESS PATH ANALYSIS, which determines how the activities use the data.

The end deliverable of this step is the logical database design. The logical database design is then used as input to the PHYSICAL DATA STRUCTURE DESIGN or PHYSICAL DATABASE DESIGN process.

In the following sections of this paper, the various steps of the Systems Development Cycle, which result in the production of the logical database design, are discussed in more detail, using the breakdown shown in figure 2.

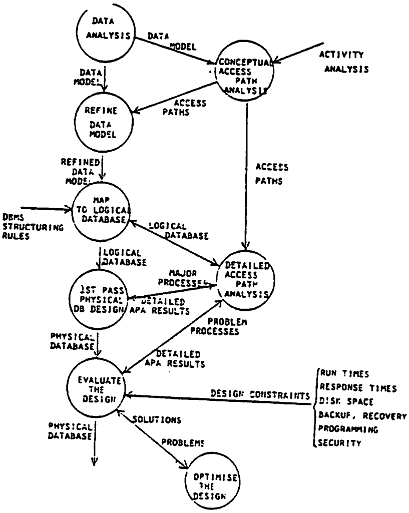

DATA ANALYSIS
DATA MODEL
CONCEPTUAL ACCESS PATH ANALYSIS
ACTIVITY ANALYSIS
DATA MODEL
ACCESS PATHS
REFINE DATA MODEL
REFINED DATA MODEL
ACCESS PATHS
MAP TO LOGICAL DATABASE
DBMS STRUCTURING RULES
LOGICAL DATABASE
LOGICAL DATABASE
DETAILED ACCESS PATH ANALYSIS
MAJOR PROCESSES
1ST PASS PHYSICAL DB DESIGN
DETAILED APA RESULTS
PROBLEM PROCESSES
PHYSICAL DATABASE
DETAILED APA RESULTS
EVALUATE THE DESIGN
DESIGN CONSTRAINTS
RUN TIMES
RESPONSE TIMES
DISK SPACE
BACKUP, RECOVERY
PROGRAMMING
SECURITY
SOLUTIONS
PROBLEMS
PHYSICAL DATABASE
OPTIMISE THE DESIGN

Figure 2.
The Database Design
Process

In this diagram Logical Data Structure Design has been broken down into two steps, namely 'REFINING THE DATA MODEL' and 'MAP TO THE LOGICAL DATABASE'. Physical Data Structure Design has been decomposed into '1ST PASS PHYSICAL DATABASE DESIGN', 'EVALUATE THE DESIGN' and 'OPTIMIZE THE DESIGN' steps. Access Path Analysis has been broken down into 'CONCEPTUAL ACCESS PATH ANALYSIS' (used to support the Logical Data Structure Design) and 'DETAILED ACCESS PATH ANALYSIS' (used to support the Physical Data Structure Design).

This discussion is followed by a detailed examination of the steps that can be taken to optimize the logical database design.

DATA ANALYSIS

DATA ANALYSIS is a method used to understand and document a complex environment in terms of its data resources.
I use here the ENTITY-ATTRIBUTE-RELATIONSHIP (EAR) method, a more rigorous form of Peter Chen's Entity-Relationship approach.
The output of the Data Analysis consists of a Data Model Diagram which describes the data resources in terms of entities and relationships, and a data dictionary describing the entities, attributes and relationships.

An ENTITY is something of fundamental importance to a company. It is thus something about which data will be kept in an information processing system. An entity is shown on the data model by means of a rectangular box, with the name of the entity inside. Examples are : objects, people, places and abstractions such as an event.

An ATTRIBUTE is a basic unit of information which describes an entity. Within the company environment an attribute cannot usefully be subdivided into other units of information. An entity must have attributes if it is of interest to the company. Examples are : policy number, date policy started, person's name, person's date of birth.

A RELATIONSHIP is an association between entities. It is shown on the data model by means of a line drawn between the entities. Examples are : the entity order is related to the entity "order-line", the entity "car" is related to the entity "part".

Whether something is an attribute or a entity is dependent on the company environment. For example in a car factory, colour is an attribute of the entity car because colour is a basic unit of information .In a paint factory, however, colour is an entity because it probably has a number of attributes.

A relationship has a degree, this indicates how many occurrences of an entity are related to one occurrence of the other entity. The degrees are one-to one, one-to-many and many-to-many.

A relationship also has a type, e.g. OPTIONAL, this indicates that an occurrence of one entity MAY be related to one or more occurrences of the other entity in the relationship.

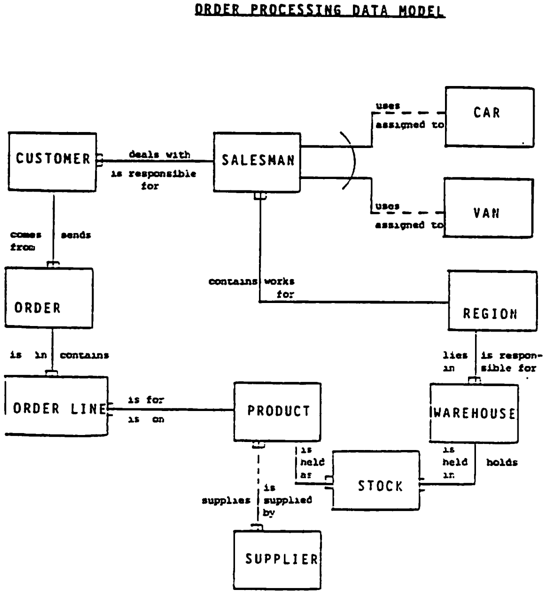

ORDER PROCESSING DATA MODEL
CUSTOMER
deals with
is responsible for
SALESMAN
uses
assigned to
CAR
uses
assigned to
VAN
comes from
sends
ORDER
contains
works for
REGION
is in
contains
lies in
is responsible for
ORDER LINE
is for
is on
PRODUCT
WAREHOUSE
is held as
is held in
holds
STOCK
supplies
is supplied by
SUPPLIER

Figure 3. Data Model

This is shown in the data model by a dotted line between the entities, an example is the relation between PRODUCT and SUPPLIER (see figure 3). It means that there can be a product without a supplier (this company also makes their own products).

Another relationship type is EXCLUSIVE; this means that an entity is related to either one of a number of other entities. For example the entity SALESMAN is related to the entity CAR or to the entity VAN, this is shown by an arc (see figure 3).

The Data Model diagram contains all entities and relationships in the scope of the project. The relationships lines can have at end special symbols as arrows or tridents indicating the relationship degree (see figure 3). In addition to the data model all entities, attributes and relationships should be documented in a data dictionary. This documentation should also include precise descriptions, volumes, characteristics, special conditions and values etc. Figure 3 shows a Data Model for an order processing system.The Data Model is used to enable the users and the information-analysts to quickly understand a complex data area (one diagram is more than thousand words). The Data Model is the basis for the logical database design.

Building the Data Model is an iterative process. By definition, the analyst does not know the details of every entity, attribute and relationship at the start of the analysis . Intermediate Data Models will be wrong, but should allow the analyst to extract further information from the user in order to arrive at a correct model.

CONCEPTUAL ACCESS PATH ANALYSIS

Before a database can be designed we must understand the data structure (Data Analysis) and how the data is used. Access Path Analysis is the process of showing how the business activities defined in the Activity (or Functional) Analysis uses the data units defined in Data Analysis.

The functions identified during the Activity Analysis can be grouped as retrieval and creation. The retrieval functions just read the data, while the creation functions may, apart from reading, also insert, delete and modify the data.

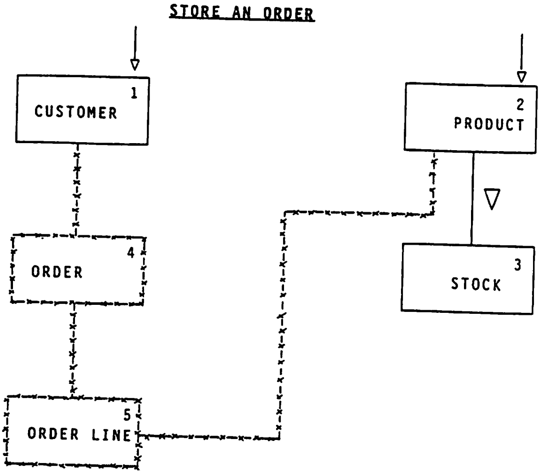
For every function a profile must be made indicating the initial access to the data and the way the function uses the data structure.

An ENTRY POINT shows the initial access to the data. It is a search for one occurrence of an entity (e.g. customer-code 159) or a search for several occurrences (e.g. all customers called Smith).

A NAVIGATION PATH shows the usage of the data structure. It consists of one or more entities and their relationships.

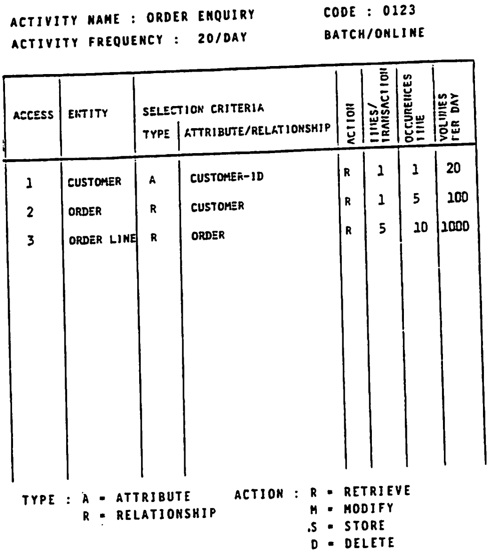
An example is shown in figure 4; here the creation profile of the "Store An Order" function is drawn. The numbers in the entities shows the order of access. The first entry point is Customer-no in the entity Customer, to check if the Customer exists. The second entry point is Product-no in the entity Product to check if the required product exists, then following the path to the entity Stock is followed, in order to check if there is stock for the required product etc.

Figure 4. Creation Profile



In addition to drawing the access profiles in the Data Model, forms should be filled in for each process showing frequency,volumes of data accessed, specific attribute usage within the entities, whether it is a response time critical function etc.(see figures 5 and 6). From these forms a number of statistics and matrices can be generated :

- Usage statistics show how frequently entities and relation are used.
- Usage matrices show in which functions the entities and relationships are used and/or created. Throughout the database design process it is essential to have an easy cross-reference of activities (or processes) to entities (or files). This is the only way to evaluate the total effect of design decisions.
- Attribute usage matrices show, per entity, where attributes are used and/or created.



ACTIVITY NAME : ORDER ENQUIRY CODE : 0123
ACTIVITY FREQUENCY : 20/DAY BATCH/ONLINE

ACCESS	ENTITY	SELECTION CRITERIA		ACTION	TIMES/ TRANSACTION	OCCURENCES TIME	VOLUMES PER DAY
		TYPE	ATTRIBUTE/RELATIONSHIP				
1	CUSTOMER	A	CUSTOMER-ID	R	1	1	20
2	ORDER	R	CUSTOMER	R	1	5	100
3	ORDER LINE	R	ORDER	R	5	10	1000

TYPE : A = ATTRIBUTE
R = RELATIONSHIP

ACTION : R = RETRIEVE
M = MODIFY
S = STORE
D = DELETE

Figure 5.
Access Path Form

ACTIVITY NAME : ORDER ENQUIRY CODE : 0123

Figure 6.
Attributes used per Activity

ACCESS	ENTITY	ATTRIBUTES REQUIRED	SORT FIELDS
1	CUSTOMER	CUSTOMER-ID	
		CUSTOMER NAME	
		CUSTOMER ADDRESS	
2	ORDER	ORDER-ID	
		TOTAL VALUE	
		BTW	
		DATE	DESCENDING
3	ORDER LINE	LINE-ID	ASCENDING
		PRODUCT-ID	
		QUANTITY	
		PRICE	

The matrices and statistics can be compiled manually but since the design process is iterative it will be a lot of work. A better solution is the use a good Data Dictionary. Most analyst-workbenches generate the matrices and statistics automatically.

REFINING THE DATA MODEL

The aim of this phase is to create a data model that contains only the required data and access structures, using the results of the conceptual access path analysis. What is required to be implemented may differ from what is conceptually present, because it is not normally practical to automate everything.

The transformations carried out in this phase are as follows:

1) Relationships

- An added relationship may be redundant in the conceptual model, but important as a result of access requirements.
- A relationship may exist in the conceptual model but will be eliminated when conceptual access path analysis does not find a requirement for it.

2) Entities and Attributes

- If access path analysis shows that an entity is never used, it can be eliminated. Similarly, attributes can be eliminated if access path analysis shows that they are never used. If you decide to leave them in, remember that they may need to be maintained. If you leave them out ask yourself the questions : will it be required later and how easily can they be added. Make sure you document your decisions.
- Entities which are always accessed together can possibly be mapped to one record type. This depends on the number of occurrences, the volume and the variability.
- Exclusive entities with similar attributes could be combined. An indicator must be added to show the difference in the occurrences.

3) Many-to many relationships

Because few database management systems will handle 'many-to-many' relationships, the should be eliminated at this stage. They are eliminated by the introduction of a new "in-between" entity (sometimes called a "relation entity") .The new entity is related to the original two entities with two 'one-to-many' relationships (see SUPPLIER-PRODUCT)

Starting from your conceptual data model you have selected everything that was required using the access path analysis results. You have simplified and optimised the required parts of the data model. You have created the REFINED DATA MODEL; now you are ready to map to a Turbo-Image database. Make sure you have documented all changes in the Entity-Attribute-Relationship descriptions.

The Refined data model is shown in figure 7.

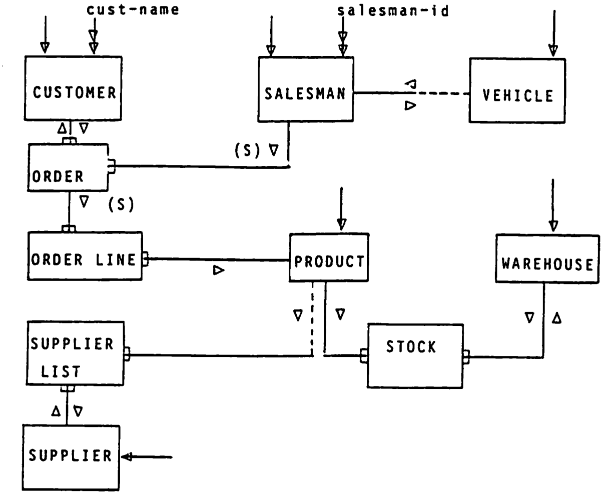

cust-name
salesman-id
CUSTOMER
SALESMAN
VEHICLE
ORDER
(S)
(S)
ORDER LINE
PRODUCT
WAREHOUSE
SUPPLIER LIST
STOCK
SUPPLIER

Figure 7.
Refined Data
Model

MAPPING TO THE LOGICAL DATABASE

The Turbo-Image database management system has a set of "structure rules" which does not allow direct implementation of the refined data model. We must therefore translate the refined model to the logical model. We must also implement the access requirements into this logical model. This step is called mapping.

The Structure Rules for Turbo-Image are as follows:

1) DATA SET

A DATA SET is a collection of records with the same record format. In conventual terms it is a file with one record type. Turbo-Image supports two types of data-sets, MASTER and DETAIL.

2) PATH

Relationships are supported as PATH's between masters and details. These path's consists of one entry in a master set linked to zero, one or more entries in a detail set. The linking is done via a common value in the search field of a master set entry and the search field of a detail set entry.

A path is a means of connecting entries in master data sets to entries in a detail data set. Paths only exist between masters and details. The search field in the master data set is linked to one of the search fields in the detail set if they contain the same value. The physical implementation of the path is called a chain, which is a pointer in one entry pointing to another entry in the same set (as in details) or in a different set (from master to detail). The entries in a chain in a detail set may be sorted on a data field in that entry. This data field is then called the SORT field.

3) MASTER DATA SET

A master data set has one unique data field called the search field. Turbo-Image supports calculated (hashed) access to the master set using the value of the search field. A master data set can be linked to up to 16 detail sets.
Two types of master sets are supported, the MANUAL master set and the AUTOMATIC master set.

4) MANUAL MASTER SET

The manual master may contain more data fields besides the search field. The master data set must be maintained by the application programs. If an entry must be added to a detail set, linked to a manual master set, an entry with the same search value must first be created in the manual master set.

5) AUTOMATIC MASTER SET

An automatic master set entry contains only one data field, the search field. The automatic master set is maintained by Turbo-Image. When a data entry is added to the associated detail set using a value of the search field that does not already exits in the automatic data set, an entry with that search value is created in the automatic data set. When the last entry with a certain search value is deleted for the associated detail set, the corresponding automatic master set entry is deleted by Turbo-Image.

6) DETAIL DATA SET

The detail dat sets can contain several data and search fields. A detail data set can be linked to maximum 16 master data sets.

7) ACCESS METHODS

Several access method are supported by Turbo-Image. For the logical database design the most important are shown grouped by data set type:

Access on masters : SERIAL - the sequence in which the entries are physical stored
HASHED - the value of the search field is used to find the entry

Access on details : SERIAL - the sequence in which the entries are physical stored
CHAINED - all detail entries belonging, via the same search value, to one master entry. The sequence of the chain may be controlled via a sort field.

For other "structural limitations" I refer to the Turbo-Image reference manual (32215-90050).

The mapping process consists of 13 steps which map the refined data model to a logical data model.

1. Select all entities from the refined data model that only have 1:1 or 1 : M (many) relationships with other entities and map them to a manual master data set.

2. For each manual master determine, from the access profiles, the search field. If more than one search field is required, choose the search field required most often for on-line access.

3. Map all remaining entities to detail data sets.

4. For every detail set, implement the M : 1 relationship that exists in the refined data model with the entities that were mapped to manual masters in step 1, and include the search field of the master and the detail data record.

5. Select all entities that were mapped to detail data sets in step 3 but have 1 : 1 or 1 : M relationships with other entities in the refined data model.
   Choose a suitable search field for these entities (refer to the access path analysis).

6. Create automatic masters containing the selected search field for the detail data sets selected in step 5.

7. Implement the 1 : M relationships that exists in the refined data model between two entities that were mapped to detail data sets by relating them to the automatic masters created in step 6. Include the search field in the detail data record.

8. Implement the relationship between two manual master sets (1 : M and 1 : 1 relationships in the refined data model) by creating a detail data set containing only the search fields of the manual masters.

9. Consider extra automatic masters for details on which direct entry is required (see access path analysis) or access via the existing master sets is clumsy.

10. Remove all paths (not the search fields) when the access profiles show that you only go from detail to master and never from master to detail.

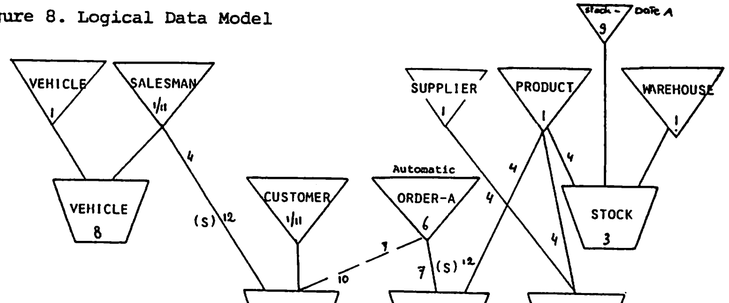
11. If more than one search field on a master set is required, consider mapping the master to a detail with automatic masters for the needed extra search fields.

12. Determine for the access profiles which master to detail paths should be sorted, and on what key. If more than one sort order is required, choose the one required most often for on-line retrieval.

13. If a detail data set has an optional relationship with a master set implement a extra automatic master and an extra detail, because Turbo-Image will not allow non-owned detail entries.

The logical data model shown in figure 8 is produced via the mapping rules from the refined data model and the access profiles shown in figure 7. The number of the appropriate mapping rule is shown for reference only.

Figure 8. Logical Data Model



1ST PASS PHYSICAL DATABASE DESIGN

The logical data model must now converted to a physical model, on the basis of the following considerations:

- performance optimisation
- environmental constraints (CPU, disk-io's, disk space)
- user constraints (response times, run times, security, backup and recovery)

This process is outside the scope of this paper.

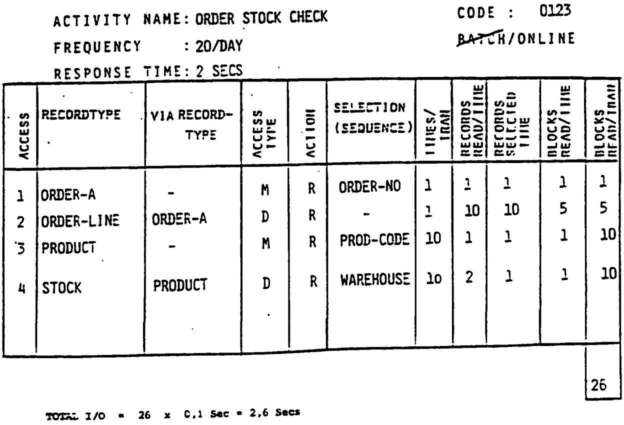
DETAILED ACCESS PATH ANALYSIS

This process supports physical database design, and optimization, by evaluating how a particular function will use the database in terms of :

- use of entry points
- records read
- records selected
- use of paths
- sorts
- frequencies

The detailed access path analysis differs from the conceptual access path analysis that it looks at volumes, importance and time-criticality.
An analysis is done for all major processes to check for problem areas in the logical model.

Figure 9. Evaluation of physical I.O.



ACTIVITY NAME: ORDER STOCK CHECK

CODE : 0123

FREQUENCY : 20/DAY

~~BATCH~~/ONLINE

RESPONSE TIME: 2 SECS

ACCESS	RECORDTYPE	VIA RECORD-TYPE	ACCESS TYPE	ACTION	SELECTION (SEQUENCE)	TIMES/ TRAN	RECORDS READ/TIME	RECORDS SELECTED TIME	BLOCKS READ/TIME	BLOCKS READ/TRAN
1	ORDER-A	-	M	R	ORDER-NO	1	1	1	1	1
2	ORDER-LINE	ORDER-A	D	R	-	1	10	10	5	5
3	PRODUCT	-	M	R	PROD-CODE	10	1	1	1	10
4	STOCK	PRODUCT	D	R	WAREHOUSE	10	2	1	1	10
										26

TOTAL I/O = 26 x 0,1 Sec = 2,6 Secs

For every major process a form, as in figure 9 for the "order stock check" function, is filled out. This form then shows, per user process, the expected database response times.

This is the first moment in the database design process that we can see the number of I/Os that a certain function is generating. We can also see if we can achieve the required response time.

EVALUATING THE DATABASE DESIGN

This phase takes the physical database design, and determines what is wrong with it. The design is wrong if the design constraints are not met; i.e. the run or response times are too long, the structure is so complicated that programmers have difficulty in using it properly, security is compromised, or the operational tasks cannot be performed effectively.

When evaluating the design, we concentrate on the problem areas. Performing Detailed Access Path Analysis takes a long time, and there is no point in doing so if there is little likelihood that the results will impact on the final database design. The following problems can be identified during the evaluation:

1. Access paths too long
2. Too many access structures
3. Wrong primary path
4. Locking too much data
5. No distinction between files and tables
6. No distinction between current and historic data
7. Large entries
8. Long back-up/recovery

The following section describes the Logical Database Optimization Solutions that can be used to address those problems.

OPTIMIZING THE LOGICAL DATABASE DESIGN

When optimizing the database design, we try the simplest solutions first. For example, physical database design decisions are re-examined, and where necessary reversed. However, the problems are often more fundamental, and, like it or not, the logical database design must be optimized. The possible measures discussed on a per-problem basis, as follows:

1) ACCESS PATH TOO LONG

A function may have to go through a far too long access path to get the required data. Solutions are to add redundant data or to a redundant access structure.

Put data items you would otherwise have to determine by accessing one or more detail records in the master record.
If particular reports are often required, calculate them once and store them in a separate data structure.

2) TOO MANY ACCESS STRUCTURES

An "access structure" is a database structuring which is used to find a record, it is any access method other than serial search or sorting. Every access structure in a database needs space and causes maintenance overhead.
Each one must be examined to determine if it is really necessary. Consider external sorts and/or program internal tables against the overhead of maintaining sorted relationships.
If a detail set has too many masters (say more than four) the number of disk I/O's it takes to ADD an entry to this detail set will probably give unacceptably long response times .

3) WRONG PRIMARY PATH

The primary path is defined as the most used path to a certain detail set. This can be determined from the detailed access path analysis. It should however be evaluated when the database is in production. The Hewlett-Packard product PROFILER shows exactly what the most used path into a detail set is.
Keep in mind that the primary path is optimal only after reorganisation of a detail set.

4) LOCKING TOO MUCH DATA

When a small subset of attributes (fields) are subject to frequent change, causing the whole entry to be locked, think about splitting the data set.

5) NO DISTINCTION BETWEEN FILES AND TABLE

If a data set has only a few, small records which are infrequently updated, make it a look-up table or a data set with no relationships. The integrity constraints must then be handled by the application programs.

6) NO DISTINCTION BETWEEN CURRENT AND HISTORIC DATA

The critical questions to ask are as follows:

- When does current data become historical?
- What happens to it when it is historical?

- How can you recall it in the future?
- Should you split a data set?
- If so, what happens to the functions or queries that need all the information available?

7) LARGE ENTRIES

Large entries will require large buffers, and this will bring down the number of buffers, thus making the database slower in multi-user interactive environments. A remedy is to split the entry (make it two data sets).

8) LONG BACKUP/RECOVERY

If your database is very large it will take long time to back it up. Normally this means that the database is not available for users and if you do the backup at night it means extra operator hours.
If you split your database in two parts and if you have two tape-units you can do the back-ups concurrently in half the time it used to take.

CONCLUSION

Optimisation is a process in which you :

- pay the most attention to parts of the database with known problems,
- try the simplest solutions first, and
- preserve the data-structure as much as possible.

Optimisation is a process of making trade-offs. The major trade-offs are between update and retrieval efficiency, and between ease of use and performance. It is important to strike a balance between the effort required to optimize the database and the return on this investment. In short, designing a 100 % optimal database is not a 100 % optimal use of our time.

ELECTRONIC FORMS IN THE HEWLETT PACKARD ENVIRONMENT

Barry Gillespie
Indigo Software Ltd.
560 Rochester Street
Ottawa, Ontario K1S 5K2
Canada

1. INTRODUCTION

This paper will discuss the use of electronic forms in the Hewlett Packard environment. An "electronic form" is a means of printing formatted business data combined with graphic elements **without using pre-printed forms.** The "Hewlett Packard environment" includes the HP Vectra PC, HP3000, HP9000-300, HP9000-800 and HP1000 with one of the family of HP LaserJet printers. These printers, (LaserJet Plus, LaserJet 500 Plus, LaserJet Series II and the LaserJet 2000), will be referred to by their collective name "the LaserJet". The original basic LaserJet is not included because it lacks sufficient memory, vector graphics capabilities, macros and font variety to make it useful for anything but the simplest of forms applications. This paper does not directly address the HP2680 and HP2688 high end laser printers, though many of the ideas presented here are applicable.

The use of electronic forms will be examined from several perspectives:

1) What are the benefits of electronic forms compared to using pre-printed forms?

2) What tools are required for electronic forms design?

3) How can the capabilities of the HP LaserJet be used to optimum advantage?

4) Are there tasks that cannot be accomplished with pre-printed forms that are now possible with electronic forms?

2. BENEFITS OF ELECTRONIC FORMS

There are many benefits that can be derived from using electronic forms instead of pre-printed forms. Some of the benefits produce direct cost savings. Other benefits are related more to operational flexibility, efficiency and productivity.

The first and most obvious benefit of using electronic forms is **reduced consumables costs** as compared to using pre-printed forms. This is especially true with the LaserJet 2000 which has consumable costs of only $0.01/page. In addition, duplex printing can cut paper costs in half. Printing data on both sides of pre-printed forms with an impact printer is not feasible.

No investment in pre-printed forms inventory is required. First, there is the cost of the pre-printed forms themselves. Payment must be made immediately for forms that sit on the shelf, sometimes for months. With electronic forms **there is no forms inventory or forms storage costs.** Finally, there is no need for staff to manage the shipping, receiving and handling of the forms inventory.

With interactive graphics forms design software **the lead time to put new forms into production is greatly reduced.** This is often of critical importance in today's rapidly changing business environment. Designing a form takes much less time if you use an interactive design tool. The savings that a draftsman can gain by using Computer Aided Design (CAD) tools has been well documented. Automated forms design provides similar benefits to the forms designer. These time and cost savings are even more dramatic when simple revisions must be made. Instead of going through four to six weeks of re-drafting and reprinting, the new version of an electronic form can be available in minutes.

The fact that operational forms are constantly changing introduces another area of cost savings. **Obsolete forms never have to be thrown out.** In most organizations, between five and twenty percent of all pre-printed forms end up in the garbage.

Electronic forms are **much easier to distribute.** They are usually simple binary files that can be distributed over normal communications links very easily. All branches of a business, whether "down the street" or "down under", can be updated simultaneously when the new forms are ready to go into production. The old version of the form can be deleted or archived. There is no possibility of users inadvertently using outdated forms.

No operator intervention is required to change forms. All form changes are controlled automatically and only plain, white stock paper is ever loaded into the printer. This means that expensive hardware resources are never sitting idle waiting for physical operator intervention. It also means there may be no need for an operator at all.

Since the operator is not required to change the forms, there is no chance they will mount the wrong form by mistake. **This reduces errors and wastage.** In addition, the scheduling of which print job runs next becomes much more flexible, since the forms mounting requirement has been removed.

When the financial and operational value of all these benefits is calculated, the cost justification for buying laser printers and the required electronic forms software becomes obvious. Financial payback in a year or less is not uncommon.

3. ELECTRONIC FORMS SYSTEMS - REQUIRED CAPABILITIES

3.1 FORMS DESIGN

There are currently two different types of tools being used to design electronic forms: code based and interactive forms design systems.

Code based tools require the designer to create the form specification in a specialized language. The code-based approach has several disadvantages. The first and most obvious is that the designer cannot see the results of his work as he goes. He must visualize it as he generates the form design, and hope that the result matches what is wanted. Often, several rounds of trial and error are required before the desired result is achieved.

Code based design tools require a skilled coder in order to be used successfully, especially if the form is at all complicated. Specifications of several hundred lines are not uncommon. Making changes to these long, complex coded forms specifications is difficult and time consuming.

Interactive form design programs solve all of these problems. With a good WYSIWYG (What You See Is What You Get) interface, the designer can see the form take shape on the computer screen as it is drawn. Wordy commands do not have to be coded. Often all that is required is a few clicks of a mouse button. Forms can be created much more quickly, with fewer test passes required to achieve the desired results.

A skilled coder is not required. The forms designer who formerly used a drafting table can quickly learn to use a well designed interactive graphics design tool. The analogy to a draftsperson moving from a drafting table to a CAD system is obvious. Corresponding increases in productivity will follow.

Many products that have been designed for Desk Top Publishing are also being presented as forms design tools. This is misleading. A product designed to handle the text intensive task of typesetting a newsletter, a long report or manual is ill-equipped to design forms. A form is usually largely a graphic entity. A good interactive forms design tool will be much more like a CAD system, with added capability of handling the much more complex text and font requirements of many common forms.

Most forms are set in a sans serif font like the Helv font available with several font cartridges and downloadable fonts available for the HP LaserJet family of printers. An interactive forms design tool must be able to represent the space taken by text accurately on the screen. Seeing the text on the screen in the actual font at the proper point size, boldness, etc., is also desirable.

A string of text with a mixture of regular, bold and italics, perhaps in different point sizes, should be stored as a single logical text item. This is important because during the life of the form words in the string may change. If the string is stored in pieces, much shuffling around will be required to implement a simple change.

Four different types of text alignment are required: flush left, flush right, centered and justified. An interactive forms design system must be able to handle all four types of text alignment -- even when the text contains multiple fonts in different point sizes. The capability of controlling the white space between lines is also essential.

A facility for handling raster images is required. Logos, signatures and small pieces of "artwork" occur on almost all forms. The ability to handle such images is essential. They can either be drawn with a paint program or scanned using a device like the HP ScanJet.

Users often ask why they cannot just scan in an entire form. A letter size (8.5 by 11 inch) form scanned in at 300 dots per inch (the resolution of both the ScanJet and the LaserJet) takes about 1 megabyte of memory. Even when raster compression techniques are applied, the scanned form will take up to thirty times the memory required to represent the same form in PCL (the printer control language used by the LaserJet). Every time the form is printed, this large raster image must be transmitted to the printer. The speed requirements of electronic forms applications usually make the production use of scanned forms impractical.

The use of scanned forms also does not allow for easy revisions. The form is stored as a single complex raster image. Individual items on the form (words, lines, boxes, etc.) cannot be modified without re-drawing the form and re-scanning.

All forms are usually drawn on a grid, typically six lines per inch vertically and ten characters per inch horizontally. A forms design product should allow the user to place all items on a built-in grid. This makes corner and T junction alignment foolproof. Items that must appear directly under one another or beside one another can be easily aligned. A single set of grid values is not sufficient. Complex forms often have multiple sections set using different grids. The fine details of text placement often require use of a much finer grid. The capability for the user to change the built-in grid settings at any time during the design of the form is desirable.

The forms design system requires high level graphics facilities like those found in a CAD product, especially if complicated forms are to be drawn. Multiple layers of zoom, pan, group moves, copies and deletions all fall under this feature category. The ability to do global changes makes experimentation with alternative form designs much easier (e.g., change all text strings in a selected area from 8 point Helv to 10 point Helv).

The power and graphics capabilities of the HP Vectra ES/12, RS/16 and RS/20 make them ideal computers for the design of forms. Some arguments could be made for using an HP9000/300 machine, but the extra cost would be hard to justify. The 12 to 20 Mhz processors in the PCs will typically be more than adequate.

Up to this point only the design of the form template, the fixed portion of the form, has been discussed. A complete electronic forms system should include the capability to print the electronic forms merged with a data file to fill in the variable data fields.

The design system must allow the user to specify where on the form the variable data fields should be printed. It should allow for fields printed in different fonts, so that important pieces of information can be highlighted by bolding or by using a larger point size. It must allow for right justification of numeric data for decimal alignment. The ability to word wrap and justify paragraphs of text data is also desirable.

3.2 ELECTRONIC FORMS PRINTING CAPABILITIES.

Several methods of merging data files with the electronic forms are necessary.

The simplest form of data file merge is where the data file is simply printed overlaid with the electronic form. The data file must be formatted correctly such that the data fields are printed in the correct locations on the form. This method is exactly analogous to printing the data file onto a pre-printed form.

Simple sequential processing should be supported. The first piece of data will go into the first field on the form, the second piece of data into the second field on the form, etc. Each data value can be delimited by a carriage return/line feed, or by some special delimiter character. If this delimiter is a comma, the Basic Data Interchange Format that most PC database packages use would be supported. The sequence for filling the fields on the form can be determined by simply scanning the form for data fields top to bottom and left to right.

This method has been used by the HP Boise division to produce the catalog of third party software products that support the LaserJet. Each page of the catalog is in reality a form, a collection of data formatted on the page along with lines, boxes and fixed text labels. The advantage of this method is that the actual camera-ready production of the hundreds of catalog pages can be delayed until the last moment. Full attention can be paid to making sure the data stored in the product database is accurate. As a final step, the camera-ready pages of the catalog are printed on the LaserJet by merging the data in the database with the electronic form.

In a data processing environment, that is common on minicomputers such as the HP3000, the data files are often in a fixed record format. For example: Columns 1 to 10 contain the Purchase Order Number, 11 to 40 contain Vendor Name, etc.. The capability to extract the data fields from their specified location in the data record and then print them in the correct position on the electronic form is desirable. Support for multiple different record formats in a single data file is also desirable for more complex forms.

The electronic form system should be able to print bar codes as well as regular text. Bar codes can be required both as part of the form or for printing variable data fields. Bar codes can be treated as a special type of font. Bar codes are composed of a simple collection of vertical lines of varying widths. The LaserJet's vector line drawing capabilities can be used. The electronic forms product must be able to handle the different formatting rules for each bar code format supported, including how to generate the standard start and stop characters and the check digits required. Data that appears on bar coded labels for shipping, receiving and inventory applications can be merged with label "forms" allowing labels to be printed, on demand, using LaserJets right on the factory or warehouse floor.

In some applications it is desirable to include graphics images as part of the data to be printed on the form. An example might be a real estate listing which includes a picture of the property listed. The capability to include graphics images as part of the data is desirable for advanced electronic forms systems.

4. TAKING ADVANTAGE OF THE CAPABILITIES OF THE HP LASERJET

The LaserJet has several features that make it an excellent electronic forms printer. It also has some limitations that should be taken into account when designing forms.

4.1 GRAPHICS CAPABILITIES

PCL (Printer Control Language) is the language of the LaserJet printer controller. A system that takes full advantage of the PCL orthogonal line drawing and box drawing capabilities can greatly reduce the size of the printer data required to print the form. PCL's shaded box capability (with multiple levels of gray scale and other shading patterns) allows the form designer to use shaded areas on the form with only a few bytes of printer data.

It has often been stated that a megabyte of printer memory is required to do full page graphics on a LaserJet. If the entire form is printed as a large raster image this would be true. A forms design program that takes advantage of the vector graphics capabilities of the LaserJet can greatly reduce printer memory required. For example, the first page of a 1040 tax form can be printed with less than 20K of printer memory (not including soft fonts).

Problems can arise if a form has graphic elements that include arcs (rounded corner boxes, circles, etc.) or diagonals, since the LaserJet's vector graphics capabilities do not include diagonal lines or arcs. These types of objects must be printed using raster graphics. A large number of these objects can significantly increase the size of the PCL file needed to print the form, and reduce the speed of printing. Keeping the radius of the corner arcs small and using only short diagonals will minimize the impact of this problem.

4.2 FONTS

Three types of fonts are supported on the LaserJet; built-in fonts (i.e. fonts that are permanently available in the printer), cartridge fonts, and "soft" fonts (i.e., fonts that initially are stored on disk and can be loaded into the printer). A good forms design tool should be able to handle all font types and allow the user to mix and match fonts from all three sources. The ability to use fonts from two or three different cartridges simultaneously is also required, in order to take full advantage of the LaserJet Series II and LaserJet 2000.

Most forms are designed using sans serif Helv fonts. In all of the printers in the LaserJet line only one built-in Helv font is available (LaserJet 2000 only). This shortcoming will hopefully be corrected in future generations of printers from HP. Fortunately, the choice of fonts available with font cartridges and soft fonts is excellent. The best choice from the list of available font cartridges would seem to be the HP92286Z cartridge. This cartridge includes 8, 10, 12 and 14 point versions of Helv as well as TmsRmsn (a serif font that can be used on forms with lengthy paragraphs of text).

Helv 6 point is common on many forms, especially on more complex forms. Some forms may even need 4 point text. Some forms require large 18 or even 24 point text for titles and section delimiters. These fonts can be easily obtained as downloadable soft fonts from HP's library or from the wealth of third party fonts and font generation programs available.

A forms design system should allow for the use of any downloadable font in standard HP format. It should be able to represent them on the screen correctly at design time and download them automatically at print time. This automatic download should be optional, allowing the user to load the fonts at the beginning of the day and not pay the price for constantly reloading fonts every time a form is printed.

Choosing the right combination of cartridge and soft fonts may take some careful planning. There is a way to avoid having to use downloaded fonts at all. HP and third party vendors will make custom font cartridges on demand, putting all of the fonts required on one (or more) cartridges.

4.3 USE OF MACROS

Probably the most important feature of the LaserJet for electronic forms applications is its macro capability. A macro is a collection of PCL commands downloaded in the printer memory, ready for activation at any time. Applications can be developed where all the forms required can be loaded into the printer as macros at the beginning of the day. At production print time the application program need only send the data. In this way maximum throughput can be achieved.

One problem that may occur with this method is that the total size of all the required form macros may be too large because of the amount of repeated raster data. These raster images could be logos, signatures, etc.. If the same 10Kb logo occurs on 30 different forms it will take up 300Kb of printer memory. Raster graphics processing is also one of the slower processes in the LaserJet.

This problem can be solved by converting the raster images into a font or set of fonts. Splitting them into groups by size, so that the font "cell" size can fairly closely match the size of the images is advisable. There are several utility programs available today that perform this conversion process quickly and easily. The converted logo font(s) can be loaded with the regular soft fonts. Each logo is now stored only once in printer memory. Font processing is also somewhat faster than raster image processing. The forms will take less printer memory and will also print more quickly.

It is clear from these discussions that no simple answer can be given to the common question, "How much memory do I need in my laser printer?". The answer will depend on a careful analysis of how many forms are required, how complex the forms are, how many different downloaded fonts are used, etc.. The best plan is to build the forms first and determine the memory required -- leaving room for future expansion. Many applications can run successfully with the base 0.5 Mb memory of the LaserJet Series II. Other applications may utilize the fully loaded 5.5 Mb memory of the LaserJet 2000.

4.4 USE OF COMPILED FORMS

Any Hewlett Packard Vectra PC, HP3000, HP9000-300 or HP9000-800 computer can be used in conjunction with electronic forms. Which computer is used will usually depend more on where the data resides than on anything else. The forms are created by the design software on the PC. Before the forms are moved to the production print environment they should be compiled. The electronic form compiler converts the form from the design file format to PCL. The compiled form also contains information about the data fields. The compiled form is a simple binary file that can be transferred to any other computer using standard binary file transfer utilities.

Forms compilation has two major benefits.

A pre-compiled form prints more quickly. A good analogy would be the difference in execution speed of executing a program from source code or from object code.

The second advantage is more subtle but of great importance to large organizations. Compiled forms allow corporate MIS and forms design groups to maintain control of the content and format of forms used in the company. The corporate form design and utilization standards can be maintained. The end users simply use the compiled forms on their production computer. They cannot change them. End users should never need or have access to the original forms design files.

The LaserJet is an ideal forms printer. Its special capabilities make it faster in a production environment than comparable printers with the same "rated" speed. It can be attached to any of the HP computers and used to produce electronic forms merged with data on demand.

5. ELECTRONIC FORMS EXTEND FORMS PROCESSING

Electronic forms allow for the processing of data in several ways that are not possible with pre-printed forms. Multiple part forms no longer need to have data fields in the same positions on all parts of the form. Forms with header and repeating detail pages can be created. Forms where all pages are not printed for a particular set of data are feasible. Finally, the template of the form can vary, depending on the data itself. The best way to explain each of these four extensions to traditional forms processing is by examples.

5.1 MULTI-PART FORMS

The separate parts of a multiple part form are actually variations on the same form. Each part has a slightly different format. For example, a three part purchase order has the three separate parts labelled: accounting copy; receiving copy; and vendor copy. The vendor copy has a special terms and conditions paragraph. The receiver's copy does not include the price and order quantity information. The degree of variation possible on pre-printed forms is constrained by the fact that the actual position of the data fields must be the same on all parts. With "electronic forms" this constraint is removed. Data fields can appear in different positions on different parts. Data fields can be printed in different fonts or even suppressed altogether.

5.2 REPEATING PAGES

Forms like invoices, purchase orders, etc. that have a large block of header data at the top followed by multiple lines of detail data at the bottom are inherently wasteful if a large number of detail lines are required. Examine what happens using current pre-printed forms technology when an invoice with one hundred items is produced. The first page contains the required header information on the top half of the form. The bottom half of the form will contain perhaps fifteen lines of detail item information. On the second page the top half is left essentially blank, except for a page number. This is wasted paper. Only another fifteen lines of detail information can be printed. This process continues until all one hundred details have been spread over seven pages.

With electronic forms only four pages are required. The first page is identical to the first page in the pre-printed form example. The second page, however, can have a different format. All the header data fields are removed. Only detail line item data appears on the second and subsequent pages. The number of detail pages used is controlled by the amount of data. In our example, the second and subsequent pages might contain thirty detail lines each instead of fifteen. The same one hundred lines of detail data will be spread over only four pages instead of seven. Continuous pre-printed forms cannot be used because the number of detail pages varies depending on the amount of data.

5.3 MULTI-PAGE FORMS

Many data processing operations require multiple page forms. In some cases, not all pages need to be filled out. A simple example of this would be an employee application form that has a page for the employee's armed services record. Only employees who actually served in the armed service have to fill it out. In all other cases this page would be left blank, wasting paper and filing space. With electronic forms this page can simply be skipped. The data file for the process can contain commands to alter the normal flow of processing and skip the unused page except when it is needed.

A more complex example is found in the insurance industry. Many policies have thirty or more related forms associated with them. Typically there is a header page of general policy information and many optional pages. Which optional forms are used for a particular client depends on which policy options are chosen. A forms processing program can have all the required forms loaded into the memory of the LaserJet as macros and then print only the forms required for each individual policy. This procedure can be controlled by allowing page selection commands to be imbedded in the data file. These commands will skip to the required page in printer memory before sending the data for that page. This type of multiple page processing cannot be accomplished using pre-printed forms.

5.4 DYNAMIC FORMS

The final example of sophisticated electronic forms processing is what might be called "dynamic" forms. It is similar to the previous multiple page example. In this case each section of data may not occupy a full page. An auto insurance accident report is a good example. Typically it will have several distinct sections; where the accident occurred, when it occurred, the weather conditions, and sections for each vehicle damaged and each person injured.

With pre-printed forms a limited number of vehicles and injured people can be entered on the form. Additional entries for these sections must be appended to the form in attached lists at the back of the policy.

A dynamic forms processor will create each page of the form based on the amount of data for each section. Sections that do not apply will be omitted. Each potential section of the form can be stored as a separate macro. Instead of placing elements of the section template in fixed positions on the page as is done with full page form macros, each section will be compiled using relative positioning. The last PCL command for each section will put the page cursor at a fixed relative reference point, so that the next section will join with the current section correctly.

Before placing a new section and its corresponding data on the page the dynamic forms processor will first determine if there is enough vertical space. If space is not available the program will eject the current page and start a new page.

Using this approach for the accident claim report will allow all vehicles involved in the accident and all people injured in the accident to be printed together sequentially. The number of pages for an accident report will vary. All data will be recorded in the required format. The dynamic form is an improved solution for this type of application.

The examples given here are some of the many ways forms applications can take full advantage of the text and graphics capabilities of the LaserJet. These same ideas can be applied to many other data processing situations in the Hewlett Packard environment.

6. CONCLUSIONS

The concept of using electronic forms to replace pre-printed forms is still relatively new. The potential is enormous. Every large business and government organization uses hundreds and often thousands of forms to control the flow of information within the business and with other institutions. The concepts presented here can substantially improve this process.

Very significant cost savings can be realized. The ability to respond to changes is much improved. The use of forms in a distributed organization is optimized. Operational flexibility is improved, with reduced errors.

In many organizations the major benefit will be that applications that cannot be implemented using pre-printed forms become feasible with electronic forms.

TITLE: Mass Storage - The Current Revolution

AUTHOR: Suzanne M. Spitzer

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 0112

The Politics of Data Communication Networks

David W. Hickey
The Ohio State University
1121 Kinnear Road
Columbus, Ohio 43212

INTRODUCTION

Many believe that the primary purpose of data communication networks is to facilitate the sharing of information. While this is certainly a function of networking, it is not the ultimate goal. The value of data communication networks is that they offer small groups, departments, and organizations a platform in which to share ideas and provide a building block for people to work as a team. By offering access to electronic mail facilities, to organizational computing facilities, and to other internal and external information resources, people can become more involved with the total organization and its' business purpose.

In any discussion on how to successfully implement a large multi-vendor data network, it is essential to remember that the goal is to integrate, not alienate, departments. By exploring technical and administrative issues as well as other relationships, we can begin to understand the political/social process that leads to successful implementation of an organizational network.

TECHNICAL ISSUES

A constant problem faced by many when attempting to establish a organizational-wide data network, is matching system design with user needs. If your company has a centralized purchasing policy that mandates what computer vendor departments must use (or in other words, dictating what equipment they perceive will meet a department's needs), then the answer is relatively obvious. You solicit the vendor for integration products that are compatible with their equipment. If on the other hand your organization subscribes to a decentralized philosophy, not unlike The Ohio State University's, in which each department has freedom to choose the computer system that best fits their needs, then the task may become considerably more difficult. This problem is compounded when your organization happens to be one of the largest universities in the country.

Consider for example that The Ohio State University's main campus, located in Columbus, Ohio, has 383 buildings situated on 1,612 acres. Approximately 53,000 students are enrolled at this campus which is supported by over 20,000 full-time faculty and staff employed by 368 departments. As a major land grant university Ohio State is so large that if it were a city, it would be among the top 20 cities in the state of Ohio!

I am sure you can imagine that with each of these departments being responsible for choosing their own computer systems, a wide variety can be found. It would be safe to say that in general most universities give true meaning to the often used phrase "Multi-vendor environment." So how do you begin to integrate all of these diverse departmental networks, the three main computer centers (academic, administrative, and health care), and the several thousand standalone PCs located at various locations spanning several acres?

About two years ago, the academic computer center decided to take the lead in establishing a high-speed campus-wide computer network, SONNET (**S**ystem **O**f **N**eighboring **NET**works). There were several valid reasons for needing to construct such a beast. First, there was an expressed need for Ohio State to become a node connected to ARPANET (a large, packet-switched network intended for the conduct of, or in support of official U.S. government business), to enhance the University's ability to communicate with other institutions for research, mail transfer, grants, etc. Second, it was becoming apparent that the campus community was being forced to devise individual ways of accessing needed resources from one another and the computer centers. If some sort of direction wasn't provided for transferring messages and data, and a standard approach wasn't found that many different systems could use, then our ability to meet the goals of education and innovation could be directly impacted.

An open invitation was extended by the academic computer center to any department (not just faculty) interested in establishing an informal study group, headed by the center's deputy director, to examine the technical concerns. From the outset it was determined that if every courtesy was not made to include any department that was interested in forming a standard way to internetwork computers, the network would fail. People do not generally embrace ideas that they do not understand or are forced upon them. To be persuasive in implementing a organizational-wide network, everyone, who may be affected, should be included.

Once established, this networking group soon realized that they needed to find a standard networking solution that was not vendor dependent. Rather than have Vendor A communicate directly with Vendor B or Vendor C, the goal would be to identify a standard medium that the majority of vendors could "talk" to. Every computer on the network could then have the same "look" as every other computer. This was important, not only from a technical point of view, but from a political perspective -- no one departmental computer would dominate and control the network. The group also believed that the criterion should provide three basic functions most important to end users: Integration of electronic mail systems; computer-to-computer file transfers; and virtual terminal capability.

Unfortunately for networking standards, they make great subjects for discussion, but if few vendors are providing workable off-the-shelf products, who wants to (or can) wait five years before they are developed? Fortunately, the standards commonly referred to as TCP/IP (Transmission Control Protocol/Internet Protocol) seemed to be the best currently available method to integrate diverse systems.

There were several valid reasons for adopting TCP/IP. First of all, it was a fairly well defined, reliable way to internetwork many heterogeneous systems. Second, it was generally associated with a larger grouping of protocols that offered the functions desired at the user level such as:

File Transfer Protocol (FTP)	Allowing a user on one host to interactively communicate with another host for the purpose of file transfer, directory listings, etc..
Simple Mail Transfer Protocol (SMTP)	The transmitting and relaying of mail, along with automatic return of undeliverable mail. The transfer of mail can be automatic or user specified.
Virtual Terminal Protocol (Telnet)	A local user on one host can become a remote user on another host by using normal log-on procedures.

Third, many of the departments' computer vendors offered software/hardware products supporting most, if not all, of the TCP/IP standards. Finally, many departments already felt comfortable with this popular group of standards. In fact, there already existed on campus an installed base of Ethernet Local Area Networks of which TCP/IP was or could be supported.

I believe it is important at this time to point out that TCP/IP was going to be the recommend networking standard. That is not to say that if departments wanted and could connect to SONNET with proprietary protocols they would not be allowed to do so. It was simply a matter that there was no other alternative but functional isolation from the rest of the organization if the majority of departments chose not to go with the recommendation. This provides quite an incentive.

Once the decision was made to go with TCP/IP, the next step was to decide on the physical configuration and interconnect methods (although it was presumed that the long-standing Ethernet specification would be the most popular way for departmental equipment to connect). Reliability and data traffic, including the amount and location, seemed to be the major concern driving the physical configuration. The general consensus was that the network needed to be based on a high-speed backbone, or data "highway", because of the enormous amount of interdepartmental information that was anticipated. To be reliable, this "highway" needed to be made of sections that could continue operating even if one section was unavailable.

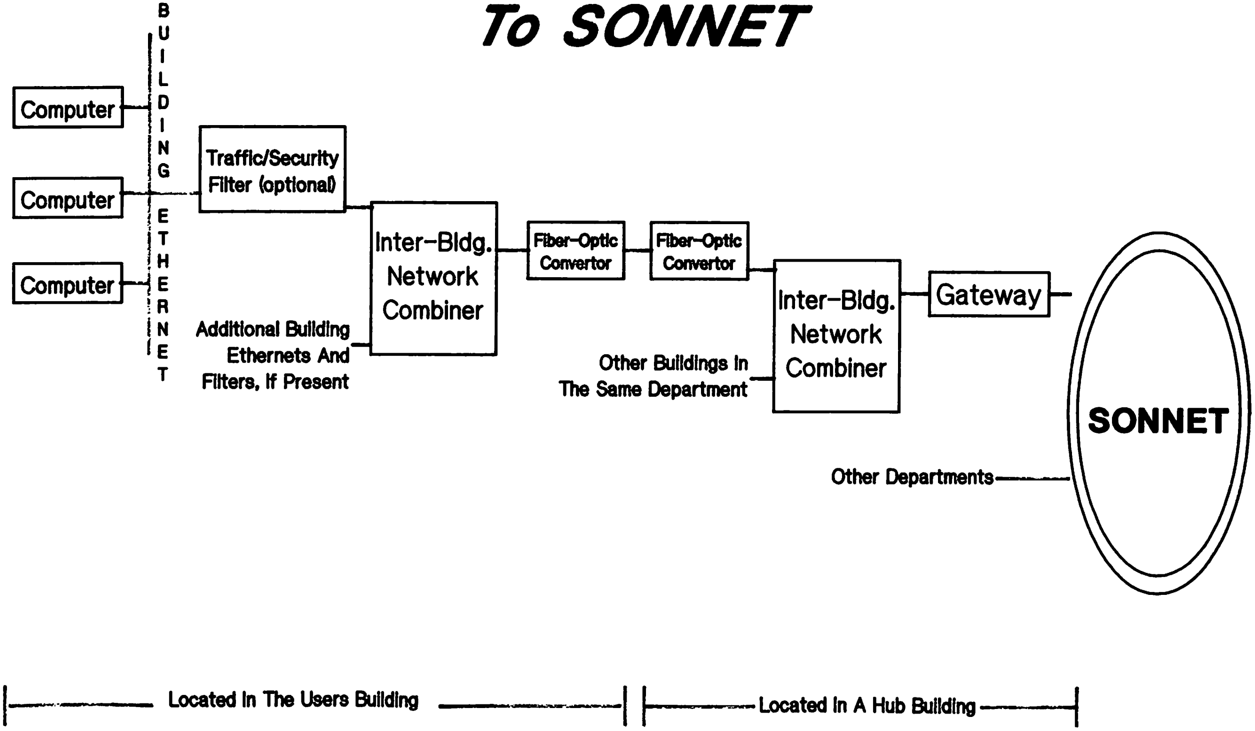
Another issue to be resolved was that traffic needed to be localized whenever possible. For instance, information sent from a local computer should not have to first be transmitted across the high-speed backbone to get to an adjacent receiving computer. Also, members were concerned that their already saturated local networks or computer systems might be used by others as a gateway to access the high-speed backbone.

The committee finally settled on a fiber-based 80 million bits per second star network (illustrated on the following page). Ethernet to high-speed token ring gateways are connected to primary and redundant fibers joining hub buildings. These help localize traffic along with other intelligent filters.

ADMINISTRATIVE POINTS

In addition to the technical issues, there are many other aspects involved in successfully implementing a organizational-wide network. The trap to avoid is to not get so wrapped up in the technical issues that behavioral concerns are overlooked. Many times the political or social process is just as important to the success of interdepartmental information resource connections as is the hardware and software that make it all work.

How Computers Typically Connect To SONNET


Computer
Computer
Computer
BUILDING ETHERNET
Traffic/Security Filter (optional)
Additional Building Ethernets And Filters, If Present
Inter-Bldg. Network Combiner
Fiber-Optic Convertor
Fiber-Optic Convertor
Other Buildings In The Same Department
Inter-Bldg. Network Combiner
Gateway
SONNET
Other Departments
Located In The Users Building
Located In A Hub Building

Adapted from a flowchart created by Instruction Research Computer Center.

- **Someone must act as the referee, coordinator, and liaison between the various groups.**

It is extremely important in the early stages of network development, when a balance is trying to be found between technical issues and political considerations, to appoint someone to act as the arbitrator. This person must be perceived as neutral and reasonably objective, and who the organization can entrust the difficult responsibility of meeting the needs of many, not just the needs of a few. We at The Ohio State University, were lucky enough to find such a person in the deputy director of the academic computer center (this opinion coming from a member of the administrative computer center).

- **Senior management must be aware and supportive of the project.**

People will be reluctant to participate if they feel that senior management is insensitive or even worse, opposed to the idea. Given the importance and high visibility of a project this size, it isn't hard to gain direct support or at least departmental participation. This is especially true in a decentralized hierarchy style of management where no one group wants to be perceived as "odd man out."

- **Develop a philosophy that will encourage the greatest number of participants to join.**

After all, the value of any corporate network, to its subscribers, increases as more members join. One obvious method to achieve participation is to mandate that all departments join which eliminates any alternatives. This usually requires stong-arm tactics -- forcing departments to abandon existing networks and join the one and true network defined as the one supported by the top political party. A more palatable approach is to gently encourage the use of the network without requiring everyone to join. By allowing the continued use of the already installed base of facilities, yet at the same time promoting the benefits of the new, user resistance is minimized. Recognize too that there will be some applications that just do not fit the "TCP/IP, etc.." mold or groups who initially do not directly benefit as members. For this reason it is very important to refer to a organizational-wide network -- not in the context of the "Fiber" or "TCP/IP" plan, but as the whole internetworked together -- because a network of data networks should serve as the balancing force in a multi-vendor environment.

- **Clarify responsibilities.**

Responsibilities need to be assigned and commitments made as to who is going to do what on a evenly distributed basis from the beginning. Unfortunately, the money is going to have to come from somewhere for installation and someone has to be responsible for maintenance: Equipment repair, assigning addresses, consulting, troubleshooting, etc. The organization needs to recognize that many of the conflicts and departmental concerns are going to be the outgrowth of how responsibilities are apportioned. If these issues are ignored in the beginning, they will later emerge to haunt the success of the project. The key ingredients rests on the willingness to have frank discussions and procedures in place to resolve conflicts that invariably surface.

- **Understand the end user.**

Users expect, and rightly so, certain levels of service. They require that the network be readily available and adaptable for use in their work environment. I have known people who use a system all day long, but when asked their opinion, they are very dejected and resentful. These problems are due, in part, to adversary relationships between end users and providers because large data concerns sometimes try to dictate what they think are the user needs. Ask questions and be prepared to respond to what the end user has to say about such issues as security, response time requirements, availability, etc.

CONCLUDING REMARKS

Technology is available today that addresses the problem of integrating yesterday's autonomous departments. By exercising care in choosing the appropriate strategy that deals with technical issues as well as social concerns, an organizational data network can provide the communication bond that is lacking in so many businesses. If a resource, such as a company-wide data network, can contribute in making each member feel that they are equal partners working towards shared goals, it has provided the essential element for any successful operation: Cooperation.

Telecommunications Management and Cost Control

Kevin C. Halvorson
Telenomics, Inc.
415 W. Foothill Blvd.
Claremont, CA. 91711

Telecommunications management and cost control issues are becoming increasingly the responsibility of the data processing professional. Upper management has traditionally looked to the data processing professional for financial data processing services such as accounting and inventory systems, to increase profits and provide cost control of company assets. It is also true that telecommunications managers are being asked to design, select, and implement data processing systems, often for the first time, in order to provide managers with more effective tools to increase company profits, while measuring costs and employee productivity.

As nations become more information oriented, corporations become more dependent on telecommunications (as a source of revenue, strategic business planning, and global competitiveness), the need to manage and control costs associated with these elements of telecommunications systems is critical to productivity and bottom-line business profits. Governments are impacted significantly as well, as revenue sources and tax bases are stressed by increasing demands. Communication management systems can be sources of revenue, measures of employee productivity, and delivery systems for community services.

Telecommunications costs are the last major expense which executive managers have yet to address with the same sophistication and management process of other major business expenses. What is critical to the successful implementation of any system are the policies and procedures unique to each organization, vertical market, and government enterprise. Often, the failure to evaluate policies and procedures as a prerequisite of purchasing a Telecommunications Management System, or TMS, sets the stage for TMS failure: for Telecommunication Management Systems do not make policy, nor can they solve inadequate procedures. Therefore, management objectives, policies, and procedures must be discussed and decisions made before a clear picture is formed of what a particular company or agency needs in order to manage their telecommunications systems.

Why is this?

> Management is generally unaware of the need for, or issues where their involvement upfront can save the organization time and money, while reducing the confusion associated with the review of alternative solutions.
>
> Management structure and reporting requirements may need to change as a result of the objectives of such a review of telecommunications strategy. Systems facilitating such changes, or consistent with existing operations, will be critical to the success of any implementation. It may also be true that management reorganization is required before a system can be brought in to meet the objectives. At the bottom line, the flexibility and support provided by the suppliers of such system solutions are critical to the ongoing success of the management process: the telecommunications environment is very dynamic, requiring constant review and change. It is demanding enough for the department responsible for telecommunications

management to keep up with their own organization's requirements, much less to keep up with the technical and regulatory changes involved in the telecommunications marketplace.

What is required for an effective solution is partnership: between supplier and end users, executive management and operational management, technology suppliers and applications suppliers. When a balance of these strategic groups and operational requirements is achieved, the overall performance, cost, and success of implementation and management of such a dynamic system are maximized.

It is the ultimate responsibility of top management to recognize the significant changes required to adequately measure and manage telecommunication costs and their impact in the organization. This process begins as a result of a company deciding on its organizational responsibilities for telecommunications:

1. voice and data responsibilities will be:
   * part of the same department **OR** * in separate departments
2. Reporting responsibilities will be:
   * same for voice and data **OR** * different for voice and data
3. Where a lack of telecommunications management experience in the organization in general exists:
   * Hire a telecom professional and establish a department
   * Identify someone to take responsibility and train them
   * Contract with a consultant to provide one-time or ongoing services:
     a. In place of a telecom department
     b. To supplement the telecom department.

It is equally important that the department or person responsible address, research, and recommend the solutions for telecommunications management, and bring them to the top of management's priorities. Some of the issues may include:

1. policies--e.g., are personal phone calls a company benefit?
2. practices-- How do we identify, charge, and collect time and money from employees?
3. procedures--Are weekly reviews of exception reports and monthly check collections required?
4. philosophies of productivity, technology and management--costs above $5 or 10 minutes are worth attention

These telecommunication management issues often do not have the focus of such management decision-making. They are critical for successful implementation and results. The primary reason is that most TMS are passive. They provide information to manage active features in a PBX such as toll restrictions, least-cost routing, authorization codes, class of service, etc.; and to enforce management policies.

What are the facts?

--Telecommunications costs represent one of the top four expenses of most organizations, often equal to data processing costs.

--Telecommunications usage costs represent one to two percent of total organizational revenues.

--Telecommunications staffs are often significantly underfunded and understaffed, when compared to data processing departments with comparable budget responsibilities.

--It is easier for an employee to make a $25 phone call than to get a $25 book on how to be a more productive employee. To illustrate this point, I offer the following:

Steps required to purchase and pay for a $25.00 book:

1. Request of supervisor
2. Purchase requisition and justification
3. Purchase order and number
4. Make purchase, and get receipt
5. Invoice to accounting, referencing P.O.
6. Accounts payable check reviewed, signed off
7. Item subject to audit
8. Item allocated to cost center.

Steps required to purchase and pay for a $25 telephone call:

1. Pick up the telephone
2. Make the call
3. Pay the bill.

There is mystery and a lack of communication between telephone carriers and customers about pricing. For example, most companies can tell you the cost of a BIC pen by color, size, and quantity purchased, with dozens of vendor options, but could not tell you the cost (with their primary vendor) of a three-minute telephone call to the top ten areas called by that company. Historically, accounting departments and firms have not understood, or had the proper tools to account for telecommunications costs. Budgets and cost allocations are either non-existent in most companies, or are based upon accounts formulas to balance G&A, versus actual usage or costs associated with telecommunications. **The telephone bill is only half the cost of making and receiving calls:** the average cost per minute of an information worker is comparable to the cost of a long distance call.

So, based upon these facts, what is the potential of internal telephone management? Significant! For instance, a pro-forma $100,000,000 company, spending 2% of sales on telecommunications, with a average profit margin of 3.4%:

> when this company saves 30% on their telephone bill, it increases bottom-line profits 18%! Using an average sales per employee of $150,000, it would take *over 117 more* employees to generate comparable profits from an increase of sales of $17.6 million.

That's why I believe that a Telephone Management System is the best investment available today for financial managers. Let's review some basics that amplify the significance of this statement. There are two types of companies or organizations:

those that are making money or are properly funded, and
those that are not.

There is a subset common to both of these:

those whose situation is getting better, e.g., making more money; or
those who are not, e.g., losing money.

The impact of an 18% profit increase through effective telephone management is critical for those companies losing money, or whose situation is getting worse. Several of Telenomics' clients purchased a TMS because a 30% savings in telecommunications costs would make the difference between profitability and a loss. It is also true that effective telecommunications plays a significant role in a company whose situation is better than our example and whose prospects are brighter. One of our clients' telephone bill has gone up every month since our system was installed. How can this be? Shouldn't a TMS save you money? Not necessarily. This company grew from 3 employees to over 3000 in 5 years. As one of the fastest-growing computer companies, they needed to make sure the telecommunications lines were available to take orders, give service to customers, and make money.

It is not the actual dollar amount that is critical in the final analysis of the success of a telephone management system implementation. Rather, it is whether the desired business results have been achieved, with budgeted telecommunications resources; whether these resources are seeing cost reduction, and whether employee use of them is being maximized for company objectives.

What are the key items on which effective telephone management can focus?
First, basic economic priorities must be set:

> The average cost of a telephone system, per extension, is about $1,000 installed. The average telephone bill generated by that extension over its life (5-7 years) is about $3,000. Assume 50% of calls are inbound with no telephone bill costs, and 50% are outbound with the associated telephone useage bill. The labor costs associated with this assumption are another $6,000 for a total costs of $10,000 per extension, 90% of which are related to time and usage. Therefore, **usage** is the first area to manage in telecommunications cost control.

Second, how are management issues grouped? There are four key areas:

- Abuse
- Misuse
- Network
- Employee Productivity

Here's a list of some of the most common uses for telephone management systems.

- *Cost Allocation*
  - Division, department, individual budgeting
- *Chargeback*
  - Clients, projects, etc...
- *Control abuse*
  - Personal calls, 976 calls...?
  - Long calls, expensive calls, competitors
- *Minimize misuse*
  - Incorrect access codes to improper circuits, excessive calls
- *Facility, Network planning*
  - Establish usage for proper mix of CA, FX WATS, OCC, TIE, T1 etc.
  - Anticipate rate changes
- *PBX Software maintenance*
  - Evaluate routing efficiencies
  - Customize class of service, toll restrictions
- *Evaluate equipment needs*
  - Plan for purchase of new PBX
  - Allocation of instruments and features
- *Improve productivity*
  - Personnel evaluation

    - Identify telemarketing areas
    - Telemarketing effectiveness
    - Collections
    - Customer service
    - Market research
    - Secretarial and message center support
- *Revenue Generation*
    - Resale of facilities
    - Redistribution of budgets from operational cost centers to telecommunications departments
- *Billing Verification*
    - Circuits working
    - Refund documentation.

Most organizations will initially identify the need for a TMS based upon a need for only a few of the above applications. Many TMS only address a few. However, once a system is installed, and the learning curve is completed on initial applications for a TMS, the need increases for the system to perform more functions, or to be more flexible. Often, these needs will outgrow the system. Telenomics' experience is that over 60% of our customers had already been using a TMS when they decided to install a Telenomics solution.

In addition to the above mentioned applications of a telephone management system, an emerging need to automate the telecommunications department functions is evolving as a requirement. The applications here include:

- Equipment inventory
- Work order
- Trouble ticket
- Cable Plant

These can be implemented in various ways. Hewlett-Packard's maintenance management package is an excellent system that many manufacturing companies already have. Lotus 1-2-3 is often used, but is limited for larger organizations where multiple user access is required. Also this approach limits reporting and integration into billing systems.

Since 1983, the phone company is no longer required to keep the records associated with the above applications. A term called USOC went away which was the standard coding system for telephone equipment. This caused a lot of development effort in systems to accomplish the above functions by telecommunications professions who were not familiar with existing commercially available inventory systems or options. Many of these new telecommunications oriented systems cost hundreds of thousands of dollars for mainframes, and are ten's of thousands of dollars on PC systems.

Based upon the cost of inventory being only 10% of the cost of a telephone extension, the inventory systems are hard to justify. They are even more expensive to implement and maintain. One benefit of implementing such a system is that you will discover that the phone company did a rather poor job. You will be surprised at the opportunity for refunds based upon a thorough audit of equipment and services.

Several consultants have specialized in providing a "split the profit" approach providing free inventory services and spliting the credits with the customer as their payament for services. Use your own judgement on this approach as compared to putting and inventory procedure in place with your own staff.

The other applications mentioned involved the automation of paper intensive systems that a computer system could improve upon with regard to speed and reporting. However, software does not solve the problems of a poor manual system and again, policies and procedures should dictate the selection of system features.

The HP-3000 office automation system HP-DESK is an excellent candidate for telecommunications departmental automation. The integrated filing, calender and electronic mail system provide a good way to communicate the multiple input for workorders, and coordination of multiple vendors for repair into a single system.

Telenomics has used HP-DESK as a transport system to communicate status of our systems operation through multiple network nodes. This gives the ability to distribute data acquisistion capabilities to reduce polling costs, yet gives a local and regional managers information as to network operation and information.

Directory systems are often part of a TMS system. They can be integrated or stand-a-lone. The HP-3000 provides for distributed directory capabilities to PC's at remote locations with centralized control at headquarters. Directories are often handled by personnel, printing department, or word processing and in almost every company and organization it is a difficult, expensive task to produce and is out of date the minute it is published. When a directory is integrated into a TMS system and controlled by the telephone operator, the first day a person reports to a new telephone extension, the directory can be updated for everyone to have access to, either on line, or in a printed report upon demand.

The newest area of telecom management is designed to increase the effectiveness of the network engineer. These specialized individuals are scarce and expensive to acquire as in-house experts. Systems that can simulate costing alternatives, traffic patterns and circuit loading options can reduce the amount of manual calculations currently done by these engineers. Network/design and optimization for simple networks can be done by relatively inexperienced telecommunication managers. Large complex networks can be designed by corporations, independent of the phone company engineering staff. The phone company staff may have a different set of priorities in designing a system than an organizations own telecom department. Here, large storage requirements, fast processing times and multiple access by various project managers are required and indicate the need for a departmental system to accomplish the day to day tasks of telecommunications engineering.

All these applications can significantly increase profitability, productivity of employees, improve service of telecommunication systems and reduce costs. However, I want to note here that it is my experience, in over five years and hundreds of prospective customer contacts that the following statements hold true:

- Companies that operate for a profit often manage their telecommunications department as a not-for-profit department.
- Organizations that are not-for-profit often operate their telecommunications departments as a for-profit department.

What causes this????

- Is it because it is no profit to commercial organizations to manage effectively their telecommunications costs?
- Is it because government agencies or departments are so desperate for revenues that any activity which provides a service deserves to be "taxed"?

I believe the above observation holds true between governmental and corporate entities. There is also another observation that can be make about companies and organizations who have not adapted to the deregulated telecommunications environment or embraced information management technologies, and those who have.

In my years of experience, the companies I have met who see telecommunications management from *where they have been*, versus *where they are going*, most often use consultants to choose the telecommunications systems to run their company. It fits best with the traditional approach of letting the phone company make the recommendation when they were a monopoly. It relieves the management group from dealing with the policy issues and business objectives of telecommunications management. However, it can also place the telecommunications manager in the role of a system operator instead of the role of a key corporate manager. It can put the consultant in the undesireable role of corporate strategic decision maker, without the *actual* responsibility of managing the implemented system or the authority to impact management attitudes in the total organization to make it effective.

Consultants can play a key role in raising organizational issues and providing training; as well as supplementing the technical expertise of an organization's staff. They should *never* be used as a complete replacement for developing in-house expertise, which can provide the added benefit of promoting teamwork among other managers to achieve corporate information management objectives using and effective TMS system.

However, the companies I have met who look *where they are going* study their needs, evaluate options, verify system capabilities, plan implementation and operations, AND make telecommunications management an active part of every managers charter in the organization. This results in effective departmental communication to top management about strategic issues of telecommunication needs, supports the telecommunication staff's requests for equipment, reinforces the role of telecommunications manager as a corporate resource; and makes telecommunications management a top management priority.

What is the significance of these two types of organizations? Primarily that the second organization is rare. The predominance of the first organization creates the development of a market for products and services that vary a great deal; which allows confusion and frustration to influence managers who make decisions on systems, and hence the importance of a TMS as a strategic tool is stifled.

- 80% of corporate and government organizations have not made policies or decisions on company practices with regard to cost control of telecommunications.
- Of the 20% who have, over 50% of them have made implemented systems to support their decisions which do not work, do not support their objectives,

or are not flexible enough to change as the company priorities and objectives change.

This means that only about one in ten companies are meeting their goals in today's telecommunications management environment. As a result of the confusion within corporate America, several companies are scrambling to take advantage of the situation, and to offer a TMS solution:

- There are many consultants in the telecommunications industry (more than 50,000 employees have been laid off since 1983);
- There are over 100 suppliers of telephone management systems; none of whom dominate the market;
- There is 20% turnover rate in company and products;
- None of the suppliers today were started by a telephone company or a leading computer company;
- Everyone claims to have a better, faster, cheaper, slicker system than the next one;
- The average system takes one to three years to buy;
- A good system installation pays for itself in less than one year.

So where do you start, in order to be one of the top 10% firms in regard to telecommunications management? You will need to start to determine how you will do the following:

1. Get top management to define objectives, policies and procedures
2. Determine the organizational impact
3. Determine the system requirements
4. Evaluate the technological options
5. Evaluate the vendor options
6. Analyze vendor pricing
7. Make a recommendation
8. Secure budget approval
9. Negotiate financing and licensing agreements
10. Implement the system
11. Provide for ongoing management of the system.

The previous list looks like a typical data processing procurement, or PBX request for proposal, checklist that most data processing and telecommunication professionals would recognize. However, the telephone management system procurement checklist is not as involved, as most systems are sold as follows:

1. What does the phone company recommend?
2. What does the switch vendor quote with the PBX?
3. What runs on a PC?
4. What can I afford to buy?
5. What does the consultant recommend?
6. What is the minimum per month cost and shortest commitment I can make?

Seasoned data processing professionals may feel a viable option is to develop this application with existing programming staff and equipment. Most MIS managers who have evaluated this option began to realize this system is difficult to design and maintain with the flexibility required to meet the needs of a dynamic industry and evolving telecommunications policies. Data-based design and maintenance issues such as changes to tens of thousands of area codes and exchanges; hundreds of tariffs and new price plans, as well as dozens of PBX formats, significantly affect the scope of the task.

It is operationally more successful, and financially more justifiable, to install a proven, off-the-shelf package, provided by a company dedicated to the support, maintenance, evolution, and operation of customer systems. Even if the system requires modifications to meet specific company needs, such modifications can be shared among other customers, or derived from participation in a users group. As the industry and user groups evolve in the implementations of such systems, companies that specialize in this system become valuable business partners. Whether it is to provide voice expertise to data processing departments, data processing experience to voice departments, or management training to executives on the implementation and operation of these systems, the benefits go beyond just software.

Even if the development cost and the application purchase price are comparable, the time lost to develop a system often is enough to justify the purchase of the off-the-shelf package. Typically, such a package is less costly to maintain because of its multiple installations, compared to a single custom installation.

It is important, whether you attempt to develop your own system, or purchase one, to understand some of the technical issues associated with the design and operation of these systems.

Technical Issues:

This portion of the paper will deal with some of the more critical and confusing aspects of telephone management system technology and design criteria. Once your organization has determined what to do with the reports available from telephone management systems, you will still have a sizable task in determining which system to implement. Additional criteria should be used to evaluate the quality, flexibility and reliability of different systems. Three basic aspects of a system must be evaluated:

1. How the data is created and captured by the system
2. How the system is maintained and processes records
3. How the system reports and distributes the information.

Data Element Characteristics:

Telephone management systems work by tracking and analyzing transaction data produced by computerized telephone systems. (PBX). These transaction records are called *call detail records* (CDR) or *station message detail records* (SMDR).

Today, there are no standards for telephone management system data. In fact, the type of data each PBX manufacturer produces varies between models, software releases, and even serial numbers. Further complicating this is frequent change of record layouts, even during a service call when a technician is working on a system doing standard maintenance. Certain data is consistent among all manufacturers and may look like this:

time	extension	#dialed	duration	trunk
9:00	2495	2134956312	02:31	6
9:01	2368	7146213395	10:35	3

There are many other types of records that exist with regard to data detail, such as trunk number, transferred calls, account codes, authorization codes, and date. There are also options which impact the total volume of records produced, e.g., inward call detail and local calls.

Data Acquisition:
The data is produced by the PBX, or by using ancillary equipment such as line scanners. Centrex systems now can provide RS-232 data output from the central office as an alternative to line scanners or costed call tapes. A costed call tape is essentially the centrex phone bill. In general, PBX manufacturers have the following options:

1. Magnetic Tape on the PBX
   This has been the traditional approach, as the tape drives were included as part of the PBX acquisition. These could easily be read by IBM mainframes, where (in the early days of call accounting) most of this sort of analysis took place. Also, call accounting started on the largest PBXs, which could justify software development for this application; hence the high-storage capacity of the tape drive was needed. The problems associated with this solution are: high cost of tape drives ($30,000 or more); high maintenance/low reliability characteristics of most of these tape drives; data unavailable until tape read (usually at the end of the month); no error reporting; manual intervention and handling of tapes causing damage, loss; and management's attention, etc.
2. Floppy disks
   With the introduction of floppy drives, manufacturers began to offer this lower-cost alternative. Most of these drives were non-standard, had substantially lower storage capabilities, higher maintenance and lower reliability than magnetic tape drives, yet they were lower in cost. These disks were then put on a special disk reader to get the information into a computer for processing.
3. Real-time data stream (RS-232 format)
   The third and best option is a pure real-time data stream from the PBX called a *list option.* For the manufacturers, it was the simplest to implement and the lowest in cost; but the burden of data capture was now on the telephone management system vendors and end-users.

In order to use an RS-232 list format, two approaches exist to provide data acquisition into a computer: *direct connect* and *data buffers*. Here the problem now becomes one of what type of computer is doing the processing of this data and what are its data communication capabilities? Buffers come in two flavors: *Solid state* (no moving parts) and *systems with moving parts* (floppy drives, winchester drives etc.) Solid state provides higher reliability, but smaller storage capacities, compared to systems with moving parts, which can provide larger storage capacities.

Computer architectures and data communication capabilities vary tremendously in the marketplace. Suffice it to say, most business computers are designed to work with terminals, tape drives, printers, etc. Communications is usually done through a front end processor or technical polling computer to do file transfers of data from unintelligent devices. When a direct connect option is chosen, a protocol buffer is often required. An example of such a product for the HP-3000 is the Telamon PBX Engine connected to a serial terminal port.

Personal computers can be programmed to provide data capture capabilities. By using serial ports, main memory, hard drives, and file transfer software to connect to mainframes, data can be uploaded for processing. Usually some intervention is required at the PC to initiate transfers. Also, the dedication of a productivity tool for data acquisition is a less than optimal investment. Most critical is the reliability and serviceability of a PC in such an application: PCs have more moving parts, boards, etc, and each component is subject to the stresses of 24-hour operation.

You will find in the market, that, as a result of experience, even the PC software manufacturers recommend a buffer in front of the PC, for increased reliability, data redundancy, data capture, ability to service the PC, and for freeing up the CPU & memory for processing.

Most mainframe equipment is weak in asynchronous communications without front-end processors and/or PC's. Netview PC is an IBM product announcement, where a PC front-end with polling software that communicates with buffers or additional PCs, uploading information to a mainframe. Now, the problem is that several devices, with multiple components and software, are required for a successful transfer of data from a PBX. This increases the opportunity for higher failure rates and makes it difficult to diagnose and repair. It also increases the cost to acquire and maintain such a system.

The previous scenario is complicated by PC communication ports and operating system limitations restricting growth. Front-end processors or technical computers lack utilities or application software, which make the system difficult to use and cumbersome to integrate into business processing applications.

Once multiple sites at various locations become part of the installation, polling over various types of data communication facilities becomes an issue. Should the data be captured using dial-up modems, leased lines, synchronous, asynchronous, LAN's, switched data, MUX, X.25, T-1 etc? Being able to facilitate existing data networks to reduce costs is important along with having proven, flexible, high reliability capability as a back-up.

Data Translation

As mentioned earlier, the data formats and information vary greatly. How can one piece of software and computer process various data from different PBXs and produce comparable results, reliably and at a low cost? There are two approaches to this:

First: Pre-processing

Pre-processing translates the raw data, in either the buffer or the PC which captures it, and sends a common format to the main computer. This supposedly relieves the software manufacturer and operations staff of worrying about format changes and such. It is a design criteria common to many software manufacturers, which the buffer manufacturers have rushed to to solve. However, as a system grows, the problem of managing the translation software, in multiple remote buffers or PCs (in either firmware *or* software), can be difficult and eventually impossible operationally. Raw records can be improperly translated and irrecoverable after transmission to the main computer. The large storage capacities of buffers can buy you time to get translation programs to the device, but the information is then delayed in being processed for use by managers, in reports or online.

Second: No pre-processing

In this scenario, the raw data is brought into the main computer without any translation, to allow for substantial error correction, redundancy of data, and control of information for translation management, at a central point. This approach with daily (or more frequent) polling allows for regular auditing of the quality of raw data, software generation, testing of communication links; as well as ease of database updates, such as area codes, extensions, and so on. It also spreads the processing requirements over a longer time, for greater flexibility in scheduling off-hours, and minimizes the impact of month-end processing requirements for data acquisition.

Data Accuracy

Data accuracy is impacted by signaling technology and pre-processing; both by the telephone company and the PBX manufacturer.

Answer Supervision

This feature communicates between switching computers when a call is actually answered, and a billing clock can start. This is currently only available to AT&T and other large vendors--not to PBX end-users as yet.

Call Record Timing

As a result of the above situations, PBX manufacturers, and even long distance carriers, set their switches to assume when a call begins, usually in numbers of seconds. Thus, the call record in the TMS will vary from the call record in the telephone bill, and the total number of records will vary as well.

Billing Problems

Due to deregulation, hundreds of new carriers came out with "dial tone", or telephone service, and hundreds of billing systems emerged. Many large carriers have purchased smaller companies, or provided billing services for others. It is therefore possible and common that multiple bills, duplicate records, and so on, **can occur**. This situation requires a highly flexible and accurate system to audit these bills, which are ultimately to be paid.

Costing Calls

Costing calls is complicated by the various ways calls are placed, rated, regulated and sold. Some costing is measured some is fixed, some are distance sensitive some are time sensitive, some are load sensitive some are circuit sensitive etc. Large carriers are regulated, smaller ones are not. Rates change with notice, others with none. The need for end users to understand how they are being billed by their carriers is the most important aspect of effective telecommunications management today. The customer orders the circuits, uses them and pays the bill.

It is not the responsibility of the software developer to provide the rates, but more importantly to provide a flexible way to implement them to maintain a high degree of accuracy. TMS system suppiers provide maintenance support services to provide rates, however, this information is available directly to the end user through the same data bases and operator services that the TMS companies use.

The most accurate method of costing is using V & H coordinates. These are the verticle and horizontal mileage distances between exchanges in North America. Other methods are average costing, zone costing, retro costing, useage costing Without using the most accurate method of costing, you will not be able to have a valid refund document, evaluate carrier pricing plans accurately, or fairly charge back cost centers. The other methods are used to simplify the programming task or to minimize processing requirements. These compromises impact the quality of the TMS system, its effective uses, and its credibility as a management tool.

System Processing and Data Base Design

System Processing and data base design for telephone management functions vary significantly, depending on the objectives the designers had, or the limitations of the operating system with which they were working. Preliminary characteristics are segmented by the following:

1. Service bureau designs, or batch processing
2. Single user/single-tasking PC, or processing on the fly
3. Single user/multi-tasking or partitioned processing
4. Multi user/multi tasking or transaction processing

Service bureau designs and batch processing are designed for operational efficiency at month end. Limited flexibility in reporting and in database maintenance are characteristics which can lead to report formats not applicable to departmental needs, as well as inaccurate data, which loses credibility for whomever (or whatever) it represents. Online inquiry is not available or very limited. If the user actually contracts with an outside third party to process, they are actually releasing valuable business information of their control suh as customer lists, volume of business etc. Another servious issue is that these systems do not meet the needs of network managers in being able to respond on a timely basis to circuit outages, new PBX least cost routing software audits and network optimization management.

Single-user systems doing one job at a time become a ball-and-chain for the manager or operator of the system. Each and every aspect of the system has to be done repeatedly at maintenance, reporting time and for daily operation. In order to provide for greater processing capacity at report time, processing on the fly is done to maximize the available processing power. This technique is often used with fragmented data records, compressed into "buckets" to save storage and speed up processing. The basic record is destroyed to accommodate the limitations of DOS and PC design.

Single user machines which partition main memory provide a multi-tasking illusion. In effect, this non-standard method of providing PC access (while buffering or processing call detail records) restricts operations on the PC and dramatically increases chances for lost data, lost files, and other errors: these systems, when loaded with file transfer software to allow uploading of information to mainframes, often crash. These methods are proprietary to most software designers, and are not documented for the end-user. Most users of such system design, after loosing call records or Lotus files, end up dedicating the PC for one application or the other, or putting a buffer on the PC to segment the applications. Even 9-track tapes have been recommended for use on PCs to solve this problem. At this point, the PC deals with telephone management as another application requiring operator intervention to execute and manage.

Multi-user, multi-tasking systems, using leading computer hardware, proven operating systems, common languages, and popular databases, are all factors which have led to the growth of distributed computing for business applications. Now that such computers are available in small low-cost configurations, departmental computing is fast becoming the growth area for computer manufacturers. In fact, the computerization of the telecommunications department is a becoming a necessity because of the shortage of experienced personnel, restricted budgets, and importance of timely information by end-user departments such as telemarketing, sales, and accounting.

The latest evolution of departmental TMS systems can be characterized by scarcity or limited installations. Few telephone management development companies chose this route in 1983, when the telecommunications management business took off, due to deregulation in the industry. Most sophisticated systems were written for IBM mainframes: IBM had a large percentage of the market and could take on AT&T. The systems which cause the most products to be announced are based around PC technology. It became available about the same time, and allowed for a low-cost way to enter the marketplace in one's spare time *or during unemployment*: remember that *over 50,000 industry people* were laid off between 1983 and 1987. Traditional call accounting system hardware manufacturers continued to make smaller, cheaper, and limited function systems, taking advantage of microprocessor technology for the mass market and vertical niches, such as hotels. Nonetheless, these systems did not address the multiple telecommunication management requirements of most business and government enterprises.

However, let us contemplate the ideal: **What if...** a telephone management system were designed with:

* An award winning database standard to manufacturers' line of business computer systems;
* A popular and easy-to-use language, such as COBOL II, where source code was provided, and a well-documented system;
* A powerful business transaction operating system allowing most system operations to be automated in job streams;
* A scalable hardware architecture, allowing a company to start out managing a single site, with growth to hundreds of sites and users;
* Integrated polling and processing;
* Multiple communications options for capturing and distributing information;
* A large portfolio of business application software for integration of telephone management information.
* Highest rated manufacturer for reliability, service and support.

What are the general designs and characteristics?

1. Home-grown software on in-house computers
2. Proprietary hardware and operating system

3. Industry-standard hardware and proprietary software
4. Industry-standard hardware and software

Home-grown software is commonly available, due to the history of call accounting and the availability of computers today. In the early 1970s, large corporations with the personnel and computer capacity, would staff projects, using the in-house accounting computer, to address specific management requests to allocate costs of telephone bills, etc. Today, with many computers available in various types for various sizes of organizations, the ability to sort call detail records quickly can be easily addressed. However, most home grown systems do not provide accurate costing and often use average, or historical bill allocation to cost calls. This results in limited usefulness do to either inaccurate reports or delays in information. Most of these systems fall short on being comprehensive by the nature of their design: they are often written to fill a specific need. Then, as needs change, or new requirements are brought up, the system has to be modified or rewritten in order to keep up. Hence systems which seem cost-effective initially end up costing many times more than an off-the-shelf system designed and maintained by a supplier dedicated to this market place.

Another critical weakness to these systems is the designer's perspective, although any system could suffer from these. Telecom people will design functional reports, but have inefficient data base design; data processing people will have great data base designs, but write reports that are not as useful. These are generalizations, but demonstrable in the marketplace.

Another big problem is turnover in high technology positions. Once the designer of the system leaves a company, so does the support: often these systems are not documented, and can no longer keep up with the changing telecommunications and computer operating system environment. These systems eventually fall into disuse, or become completely inaccurate and relatively useless.

Proprietary Hardware and Software: these systems have played a traditional role in the Telephone Management marketplace. The companies providing these systems supply them to AT&T and the regional Bell Operating Companies. Several of these companies almost went out of business due to their dependence on a few large contracts with operating companies, without adapting to the new PC open-architecture environment. However, because of the dozens of manufacturers vying to fill market niches, these devices will continue to be sold where singular functionality and low cost are required to meet a specific application. As mentioned earlier, a hotel check out system would be and example.

Another example of this is the PBX manufacturers' attempt to provide call accounting data processing concurrent with voice switching. It is typically the case with most large PBX manufacturers that they have failed miserably in their attempt to merge data processing with voice switching processing. Even the data switch manufacturers have not been successful in providing network management packages for anything other than their own proprietary system. Some new or smaller PBX manufacturers are attempting to differentiate themselves with call accounting as a feature, but again it is limited to their own products.

The primary reason for their failure is that the R&D dollars go into switching applications, rather than management applications. Even the PBX operating system will always put an MIS function as a lower priority compared to providing dial tone, or other basic features. Economics drive this as well, because the MIS function represents only about 10% of the purchase price of the PBX, and no PBX manufacturer is going to lose business over a secondary function that would make the system too expensive. An exception to this is automatic call distribution (or ACD) switch manufacturers, who have learned that the MIS function is as **important** to the sale as the reliability and features of the ACD switch.

Industry-Standard hardware and proprietary software and Industry-Standard Hardware and Software: These categories can be debated on definition alone, but this paper attempts to indicate differences in approaches to developing and marketing software that exist today. Major computer manufacturers, such as IBM, HP,DEC, UNISYS, etc, can be considered "industry standard" with their tens of thousands of machines installed in business applications worldwide.

Here the question becomes one of portability and maintainability: some developers focus on maximizing market penetration, ensuring that the software application they write will run on many manufacturers' equipment. This portability increases sales potential, but increases the costs and complexity for the support organization, who must keep the product up-to-date on current operating system releases by each supported manufacturer. Most companies providing this development and support do not have access or experience in depth in-house to provide a high level of remote training or support, on multiple manufacturers' models.

Some developers stick with proven hardware, but provide custom databases, unique calls to the operating system specific to a given model of computer, and not available to other models or compatibles; or they may include a piece of hardware, and some software, to modify the operation of a system in order to accomplish design goals. These systems look dedicated to a manufacturer, but typically are non-standard implementations requiring thorough documentation and constant developer support in order to assure reliable operation and maintainability.

A third approach is to develop and support an application on a major supplier's hardware platform; use standard database management systems, that benefit from other utilities and application packages written for the system and minimize the use of proprietary software. This reduces maintenance and support costs and allows for a rich development environment for a strong users' group.

Any of these approaches can provide users with reports. What becomes more critical with time is how the system will be supported, maintained, and enhanced, in order to meet changing industry requirements and company needs. The approach used may be more significant in the **long term** success of the system.

Reporting and distribution of information: the significance of this section is that a passive management system requires active management. This means that access to information is the most critical aspect of the system. Multiple managers, at different company levels, with different objectives and reporting cycles etc., will need access to the information., both in a facility and at remote locations.

A PC limits access to the telecommunications manager. He or she become a slave to the single-user machine responding to multiple requests to produce multiple reports. The telecom manager is often forced to limit access to the information, as a result of this sort of an implementation, and force standard reports, limited historical analysis, and limited detail. Slow print timesnd processing times, along with limited software designs and operating systems aggravate this situation. As a result, most PC implementations that are successful are in small installation with less than 200 telephones.

A mainframe environment is best used for very large amounts of data. Tens of millions of call records per month is where these machines are best used. However, because of corporate accounting systems residing on these machines, call accounting has often ended up in this environment. The problem cited most with this environment is flexibility, priority and turnaround time. Large corporate centers have production schedules, standard operating procedures etc. which limit ad-hoc reporting or on demand information. Month end processing or other peak periods for major corporate applications will put telecommunications reporting lower on the priority list.

Cost is another factor. CDR, SMDR processing can take up alot of storage and processing power. As a result, a large percentage of the overhead associated with the large corporate mainframe will be charged to telecommunications. With the power and cost effective storage devices available for mini-computers today, this application can be more effectively run on a mini-computer.

A mini-computer provides the power of a mainframe at PC prices. Its multi-user multi tasking operating system allows for many managers to have access to various modules and reports available on the system. The multiple communication cababilities such as:

serial direct connect ports to terminals
serial direct connect to PC's
LAN connections to PC's
Dial in or out
IBM or DEC mainframe links
X.25 links

These communication hardware and software options make minicomputers, expecially the HP-3000 especially suited for departmental telecommunications management.

Reports can be processed on high speed printers, downloaded to PC's or other HP-3000 for printing, provided on-line, or shipped through electronic mail. Report writers available with database management systems facilitate custom reporting requirements. Desk top publishing and cooperative processing with PC's such as products like NEW WAVE open up new opportunities for increased automation and effective communication of telecommunications information.

How Hewlett-Packard addresses this market

Hewlett-Packard addresses this market in several very important ways.

1. Hardware reliability. Most PBX's are designed for very high reliability because a telephone system that is down impacts everyone. HP-3000's track record of high reliability, coupled with features such as power failure auto restart make it a perfect match as an applications processor for the PBX. Also, many PBX rooms are not computer rooms, and the new HP Micro 3000 machines can handle a common PBX environment well.

2. Hardware support. Many telephone systems are operated 24 hours a day 7 days a week. The ability to have fast response times, world wide, with various maintenance options depending on how a company does business is critical to total uptime of a system. Proprietary hardware, PC systems etc can not match such levels of service. When HP is compared to the largest manufacturers, it comes out #1 again.

3. Software Vendor evaluation and support. HP has the #1 Software supplier and VAR program in the industry. It is designed to promote the highest standards of product development, customer support and vendor relations. The referenced software program or VPLUS is a valueable seal for both developer and customer. The high qualifications to be a Value Added System Supplier with Hewlett-Packard are another indicator of a companies dedication to a verticle market and partnership with Hewlett-Packard.

4. Technology, Communications & Office Integration. HP's commitment to improving the HP-3000 product line, increase its price performance, offer multiple communications capabilites, and provide integration tools into office applications such as HP-Desk relieves the application developer from spending resources on system related expenditures, computer environment requirements, expanding office automation demands.

In summary, a partnership between the applications developer and HP, HP and the End user, the vertical market support company and the telecommunications department, provide a strong foundation for the data processing manager, telecommunications manager and executive manager to prepare a telecommunications management framework designed to meet the goals and objectives of a deregulated telecommunications marketplace, and an information oriented business and government environment.

Management By Standards

Baron E. DeKalb III

3M Company

P.O. Box 5517

Greenville, South Carolina

Webster's defines a standard as "...something established by authority, custom, or general consent as a model or example.., something set up or established by authority as a rule for the measure of quantity, weight, extent, value, or quality...". In addition, Webster's defines standard operating procedure as "...established or prescribed methods to be followed routinely for the performance of designated operations or in designated situations...".

Using these definitions, we as data processing managers and professionals have to establish "standards" by which we manage both the operational and developmental environments that we control. By the establishment and enforcement of an environment that is conducive to a set of standards, we provide both a stable and efficient work environment for the operation of a data processing department.

To help in the establishment of standards, the following questions must, at a minimum, be addressed:

Why bother the standards at all?
In what areas should standards be implemented?
What are the cost associated with the establishment of standards?
What makes standards effective?
What are the costs of not having effective standards?
Where do I start?

In this presentation I will try to help you answer the above questions, and present a base from which you can build to establish an effective set of standards for use in your data processing departments.

To build sound systems, a guideline or standard must be adhered to so that the system has a sound foundation on which to grow. The latest trend in finding a solid foundation for systems is the use of what is called an architecture. To provide this footing for the building of systems, we must use a building block concept. By using this concept, the foundation that we use will determine the soundness of the systems built upon it. This includes the current system in development and all future systems.

Architecture is composed of five major components:

Applications
Data
Communications
Technology
Infrastructure

I have had the concept of architecture explained to me as being somewhat similar to standards. If we use this description, we are approaching the area of having our planning pay off in providing us with an architecture on which to build our future systems. By establishing standards in these areas we will provide the solid foundation required to build systems that will be easy to build and easy to maintain. As the cost of maintenance of systems increases, we must be ever aware that the systems we build will some day be modified and replaced. The business environment being so dynamic, we must provide ourselves with the tools to provide quick solutions for our organizations.

Why bother with standards at all?

As we progress through our chosen field a proven fact is that to really help in this progression we need to effectively train the person that will replace us, this will provide the opportunity to accept additional responsibilities and duties. The best way that I know of to allow this to happen is to effectively layout the requirements of the operation and programming duties that we now have responsibility for. With these areas documented in the form of easy to use and understand standards, our ability to grow is greatly enhanced.

With the ever increasing reduction in the cost of hardware and the ever increasing rise in software and its maintenance, we must find ways and/or tools to help in the control of these costs. If the effective application of standards in these areas can be effective, then we can not afford not to use these techniques to help reduce cost and increase productivity.

In what areas should standards be implemented?

The major areas for the implementation of standards are:

Operations
Programs
Jobs
Documentation

In the area of operations, we need to look at establishing standards in the following areas:

Job Scheduling

Report Distribution

Tape Handling

Disaster Recovery

There are utility programs available to help automate and control some of these activities, but if there is a lack of understanding and established guidelines are not followed then even the most versatile program will not help.

In the area of job scheduling, I recommend a daily list of the jobs to be run listed in order. Keeping a daily activity log and making this list a part of it will provide the ability to do analysis of the productive jobs run on the system. Using this or a similar method the gradual slowing down of the system can be monitored on a daily, weekly or monthly basis so that changes can be noted and corrective action taken before the situation goes from bad to worse.

A part of the documentation of each stream file should be a list of the reports generated from each along with the distribution of these reports. Nothing is worse than filling in for the operator and not being able to distribute the reports as required. In addition, this distribution list should be monitored and updated every quarter. With the dynamics of business, the report may be replaced, or the people receiving the report could either change jobs or pass the responsibility on to others.

Tape handling is a subject that we can not afford to forget. In brief, the considerations in this area are; retention, labeling, tape verification, rotation and storage.

Depending on your business, the retention of tapes can vary, but I try to follow these guidelines; retain weekly tapes for a month, monthly tapes for a quarter, quarterly tapes for a year, and yearly tapes indefinitely. The labeling of tapes should be such that the contents of the tape, the date the tape was created, the operator, tape format, machine the data is from and the retention period of the tape are all clearly defined on the label.

Tape verification and rotation go hand in hand. As the age and usage of the tape increases, its rate of error probability increases. By monitoring this, we should be able to reduce the likelihood of including a bad or marginal tape in any of our long term retention tapes. Tape storage will also effect the physical characteristics of the tape. Be sure that the temperature and humidity of the storage area is maintained as closely as possible to the actual operating environment.

The last point in this section is near and dear to the hearts of all programmers and data processing personnel, disaster recovery. As remote as the possibility of needing a disaster plan, not having one or having one and not having everyone familiar with it is only inviting chaos after the disaster. If not documented and understood by all parties involved, a data processing department without a disaster plan is a disaster waiting to happen.

I feel that the following list of information about programs will provide a basis to build a solid system with the added advantage of being able to do reports in such a way as to facilitate easy changes as systems progress into the future.

Program Location
Source Code Location
Program Type
Files Accessed
Type of File Access
Date Written
Author
Date Modified
System Documentation
User Documentation

Using this technique, if this information is kept in a database, you can run a report that list all the programs that access a certain data-set in a data base and have an idea of the magnitude of a proposed change. The next major area of standards implementation is programs. Whether you are in a third and/or fourth generation environment, having all of your programs and procedures following the same standards and convention will pay untold benefits in the unlikly event you might need to make a change to some code.

I mean standards in programming to the point of standard names for screens, data set items, working storage items, paragraph names, calculation fields, display fields, the list is near endless, but if a standard is established and followed then the writing of new programs or the maintenance of old programs can be accomplished with ease and reliability.

Let's not stop with just the internals of programs in the topic of standards. In addition to those already listed let's include screen layout, error message handling, report layout and report headings. If standards are used in these areas, then the overall design and user acceptance of the system has got to be enhanced.

The documentation of jobs is fairly simple but can greatly aide in understanding by both people not familiar with the job and will help in the orientation of new personnel. I found a good basis for the standard for jobs to be that the jobs are broken down into steps required to preform the functions to either produce the report or update the necessary files. If the steps are separate, then each part of the job can be started over in the event the job does not complete.

Just think, by establishing and using standards in these areas the amount of programmer resources that can be saved. " Yes, but I have already started my development", you say. "If I stop and start over I loose 6 to 12 months of time." I know, I have been there myself. "In this case the incorporation of standards may be worse that not having any", you say.

After the process of establishing the requirements, technique and tools required to develop the standards in the operation of our departments, next comes the ever present management question; "What is all this going to cost?". My best answer to this question might be that not doing what is necessary may cost you a promotion, or even worse your current job. With the cost of programmer time these days, a cost reduction even in the 5% to 15% range can pay back a lot of dividends over a short period of time. Taking into account only development work, there are large savings to be had but when talking about maintenance to existing programs, the savings could be even more.

The next question is one I ask myself often in situations where I do not know which way to turn; "Where do I start?". If you are just starting, start at the beginning, if you are in the middle, start in the middle, but if you are nearing the end start next time. Hopefully you have developed a technique to help develop standards in your shop, but if you have not, start now! Developing and using standards in the operations, development and maintenance areas of your shop will provide good code that is easy to read, quickly written, easily maintained and will help in the development of quick orientation of new people. I add this last comment because I am sure a few, just a few of you, might have a little turn over in your shop.

The material I have presented is not new, leading edge or earth shattering, but hopefully will bring to light the need of such standards in your department. The pay backs you can expect from your time spent in the development and implementation of these standards will provide you a solid foundation to support the rest of the development of your systems architecture.

Managing HPDesk Notice Board Over Several Nodes: Remote or Local

Bob Myers
The Ohio State University
Columbus, Ohio 43212

ABSTRACT

The use of multiple mailnodes -- and hence multiple Notice Boards -- presents a problem of providing the arbitrary user with the ability to share a 'notice' with others not on his or her mailnode. An HPDesk user does not have direct access to Boards other than the one on the same mailnode.

Procedures developed at The Ohio State University (described herein) allow any user to post Notices on any/all of 6 Notice Boards located on four 3000 processors. Functionality includes automatic removal after a week, early deletion at the request of the author and confirmation of postings/deletions from each mailnode.

Several HPDesk scripts are used to send messages to and from "Notice BOARD" users installed on each mailnode, having noticeboard capability. Processing is done via scheduled batch jobs, some of which are scheduled over DS-lines. The HPDesk Administrator needs only to monitor the Boards when a message received advises of a possible problem, or when there's suspicion of something posted in 'bad taste.'

Managing HPDesk Notice Board Over Several Nodes: Remote or Local

Bob Myers
The Ohio State University
Columbus, Ohio 43212

OBJECTIVE

Provide HPDesk users with a means to post Notices on any of several Notice Boards at multiple locations.

RATIONALE

Satisfy the need to distribute information widely without

maintaining lengthy distribution lists

or,

impacting the database with messages sent to numerous persons

Managing HPDesk Notice Board Over Several Nodes: Remote or Local

Bob Myers
The Ohio State University
Columbus, Ohio 43212

DESCRIPTION

... for the USER, scripts provide:

- easy access to his/her Notice Board
- means to get text to Boards of choice, including immediate explanation of those choices and other policy
- means of removing Notices from all the Boards

... to facilitate the process, scripts:

- execute during batch jobs scheduled throughout the day
- post Notices, schedule them for removal and delete Notices on request or according to schedule
- recognize 'priviledged' postings and handle them accordingly
- inform the submitter, by sending notification of postings, removals or problems with requests
- notify the HPDesk administrator of problems

Managing HPDesk Notice Board Over Several Nodes: Remote or Local

Bob Myers
The Ohio State University
Columbus, Ohio 43212

ENVIRONMENT

Numbers of:

Sites = 4 (Columbus, 3; Wooster, 1)

CPUs (3000's) = 5 (Two 70's, one each of 68, 42, XE)

Mailnodes = 7 (6 are Notice-Board-automated)

HPDesk users = 700 (approximate)

Max. mailnode = 285 (users)

HPDesk Administrators:

For the four Columbus sites, there is one person (with student assistant) monitoring the 6 Notice Boards on four processors. There are, however, additional persons who add and delete users. Locally-written commands (scripts) are installed, as are help screens, to enhance the HPDesk product.

At the 'distant' site (Wooster), administration is independent. At this writing, that site has not yet committed to automating its Notice Board according to the procedures described herein.

Managing HPDesk Notice Board Over Several Nodes: Remote or Local

Bob Myers
The Ohio State University
Columbus, Ohio 43212

THE SITES

Computer Center
- serves the Administrative offices of the University.
- provides electronic mail access to its customers and provides them with access to other administrative offices.
- has two 3000's with 525 HPDesk users
- uses two mailnodes for Department personnel
- uses two mailnodes for customers
- requires its employees to use HPDesk

Two other Columbus sites
- has one mailnode per location
- uses HPDesk to reach Administrative Offices

Wooster location
- uses network to reach Campus Agriculture and Administrative offices
- is not "Notice-Board-Automated"

(Notice Boards enable communicating status and schedules about computer runs, general University Announcements and other notices to widespread audiences without loading the database with messages to large numbers of persons.)

Managing HPDesk Notice Board Over Several Nodes: Remote or Local

Bob Myers
The Ohio State University
Columbus, Ohio 43212

THE ENTITIES

ON EACH 3000 PROCESSOR	... jobs scheduled at timely intervals, perhaps every 75 minutes, to do the processing
ON EACH MAILNODE	... a Notice BOARD user with "noticeboard" capability
	... public distribution lists for sending to other nodes (i.e., to the Notice BOARD users and to Admin. accounts)
	... an item 'hidden' in a Board for a general-use command
WITHIN EACH COPY OF HPDESK	... three general-use scripts
	... four scripts for Notice BOARD users
FOR EACH NOTICE BOARD USER	... three folders in Cabinet: "Purging Folders", "Folder Holder" and "Package-to-Post"
	... WorkArea text for generating messages (8 items)

USER FUNCTIONALITY

"NEWS" Command	- provides easy access for getting the user to his/her Notice Board
	- eliminates the need for new users to know where the Board is, and how to open folders...
"NOTICE" Command	- prepares appropriate number of messages for sending item-to-be-posted to the desired mailnodes for posting
	- prompts user concerning Boards on which the posting is to occur; informs what mailnodes are available
	- informs user that the item will be deleted in a week; tells how he/she can delete early
	- ensures that the item is of TEXT type
"DISCARDNOTICE" command	- must be invoked within the user's Board
	- ensures the user = Board-item's creator
	- sends a message to all Boards, so that there will be attempts to delete the item from each (that message carries a unique subject; the subject of Part 2 (text) matches the title of the item to be deleted)

NOTICE BOARD PROCESSING - 1 OF 2

Batch jobs are scheduled throughout the day. The Notice BOARD user at each node signs-on to execute four (script) commands. The first two are described below. They are daily setup commands; hence, they're exited immediately if they had been successfully executed earlier the same day.

"VERIFYNOTICES"
Command - if today's Purging Folder exists, terminate
- check that each (non-privledged) posted item is in some Purging Folder (i.e., scheduled for delete)
- if not, schedule for deletion today and send message to submitter informing of action

"SETDATE"
Command - if today's Purging Folder exists, terminate
- purge week-old Notices; inform affected users of delete, providing copy of removed item
- attempt to do same for 8-day-old Notices (in case system was down previous day)
- setup today's Purging Folder

The two scripts which do the posting and deleting of the Notices sent to the Notice BOARD users are described on the next page. They are executed about every 75 minutes during prime time.

NOTICE BOARD PROCESSING - 2 OF 2

"POSTMGMT"
Command - clean-up if previous execution of this failed

- process InTray messages as follows:
  * REPLY's are deleted
  * DELETE REQUESTS are left in the Tray
  * other messages are opened and processed according to item-type of its Part 2
    - TEXT is posted on the Board; confirmation is sent to the submitter; if 'privledged' item, old version is removed; else, present version is scheduled for delete
    - PACKAGES are replenished if subject matches existing package; otherwise, sent to Administrator
    - all others sent back to submitter

- statistics are printed on activity

"DELMGMT"
Command - upon finding a DELETE REQUEST in the InTray, obtain subject and creator from its Part 2 to match some Notice Board item; if found:
  * delete from Board
  * notify sender of removal
  * delete from first Purging Folder in which it's contained.

  purge DELETE REQUEST from the InTray

- leave all other messages in InTray

BATCH JOBS -- BOARD PROCESSING

Sample - scheduled every 75 minutes from 6 am to 11 pm

```
          !JOB BOARDSET,BOB.MYERS;OUTCLASS=LP,1
          !RUNDESK BOARD,NOTICE/COLMBS/US
          password
   ▶      VERIFYNOTICES
   ▶      SETDATE
          WORKAREA
          COPY "CONFIRMATION" TO CLIPBOARD
   ▶      POSTMGMT
   ▶      DELMGMT
          INTRAY
          9
          YES
          BOARD,NOTICE/COLMBS/OT
          password
   ▶      VERIFYNOTICES
   ▶      SETDATE
   ▶      POSTMGMT
   ▶      DELMGMT
          INTRAY
          8
          !EOJ
```

NOTES:

** the "YES" is answering the question about clearing the ClipBoard

** ergo, the "COPY ... TO CLIP" is to ensure the ClipBoard contains an item

** going to the InTray before exiting eliminates being asked if you wish to go there to see new mail

Managing HPDesk Notice Board Over Several Nodes: Remote or Local

Bob Myers
The Ohio State University
Columbus, Ohio 43212

ITEMS REQUIRED WITHIN NOTICE BOARD USER

WORK AREA of NOTICE.BOARD

Item	*Subject*	*Type*	*Created*
1	CONFIRMATION	TEXT	04/27/87
2	CONFIRMING DELETE	TEXT	02/19/88
3	DELETE NOTICE	TEXT	03/18/88
4	SETUP PROBLEM	TEXT	05/29/87
5	INCORRECT PROCEDURE	TEXT	06/09/87
6	PURGING FOLDER MISSI	TEXT	10/09/87
7	UNAUTHORIZED PKG FOR	TEXT	10/15/87
8	Please Understand	TEXT	01/22/88

FILING CABINET of NOTICE.BOARD

Item	*Subject*	*Created*	*Content*
1	Incoming Day File.	04/23/87	0
2	Outgoing Day File.	04/23/87	0
3	Public Distribution Lists.	10/07/84	13
4	Notice Board.	10/07/84	14
5	Waste Basket	04/27/88	0
6	Purging Folders	05/29/87	7
7	Folder Holder	05/29/87	3
8	Package-to-Post	02/19/88	0

Folder.
Subject: Notice Board.

Item	*Subject*	*Type*	*Creator*
1	Using this N.BOARD	FOLDER	BOARD,NOTICE
2	...	...	...

Folder.
Subject: Using this N.BOARD

Item	*Subject*	*Type*	*Creator*
1	HowTo Read Folders	TEXT	BOARD,NOTICE
2	Getting to Notice Bo	TEXT	BOBMYERS,DESKMGR
...		...	...
n	Delete ITem#	TEXT	MYERS,BOB

Managing HPDesk Notice Board Over Several Nodes: Remote or Local

Bob Myers
The Ohio State University
Columbus, Ohio 43212

INSTALLATION STEPS

For each HPDesk Installation:

- modify scripts before installing
  - * revise node descriptions in NOTICE command dialogue
  - * edit nodal parameters (4 commands)
  - * revise name(s) of HPDesk Administrator
  - * adjust time items on Board (optional)
  - * recode handling of 'priviledged' items

- install seven scripts

- install Notice BOARD user with noticeboard capability

For the Notice BOARD user, each node:

- enter 8 TEXT items in Work Area, editing for node

- create 3 Cabinet Folders
  - Purging Folder (initialize)
  - Folder Holder (initialize)
  - Package-to-Post (empty)

- put 'priviledged' Board items at top of list

- insert special "Delete ITem#" text in some Folder
  (may need to edit DISCARDNOTICE command)

- create public distribution lists (at least 2)

Within some MPE account:

- edit jobs for signing on to the Notice BOARD user

- Schedule the jobs periodically throughout the day

For promotional or didactic reasons, optionally install:

- HELP screens

- Welcome message

- Notice on the Boards themselves

Perform Testing ... perhaps on a temporary mailnode

Bob Myers
The Ohio State University
Columbus, Ohio 43212

CLOSING COMMENTS

User complaints

- dislike name of "DISCARDNOTICE" command
  (ANS. wish to avoid one that has abbreviation coinciding with other commands; e.g., DELETE)
- do not like waiting to see my Notice on the Board
  (comment: this was only an initial complaint)
- confirmation messages from ALL Boards is confusing
  (ANS. it is believed the benefits outweigh the objections; the text of these messages was revised to offset much of the confusion)

User commendations

- appreciate policy of automatic clean-up
- like convenience of the "NEWS" command
- appreciate "NOTICE' command informing about nodes
- regard HELP screens as crucial

Desk Administrator comments

- The effort 'grew' as more ideas surfaced; hence there is some inefficiency in the code (but inconsequential)
- The overall set of procedures is now 'solid'; we would not want to do without them! We regard this as a successful endeavor.

SAMPLE SCRIPT - DISCARDNOTICE

```
&COMMENT               ..... DISCARDNOTICE Command .....
&COMMENT               Verify in Notice Board
&FORWARD 3 <NOT <AND <EQUAL <TRAY> 6> <EQUAL <AREA> 11>>>
&FORWARD CHECKPARM <REF <FIND "Using this N.Board" "" 1 -101>>
&COMMENT               Failed tests for not being in Notice Board
&PRINT  This command is executable only in the Notice Board, To
&PRINT  get to the Board, enter "NEWS".
&EXIT
&COMMENT               Check for numeric parameter with value
&COMMENT               between 1 and the 'last'.
$CHECKPARM
&FORWARD THREE <AND <PARMPRES 1> <EQUAL <PARMTYPE 1> 4>>
&PRINT This command requires a NUMERIC item number.   <CHR 7>
&EXIT
$THREE
&SAVE ITEMNO <PARM 1>
&FORWARD FIVE <REF <VAR ITEMNO>>
&PRINT
&PRINT The number you specified does NOT refer to a valid item in the
&PRINT Board.   Enter 'LIST' to see the valid items; then try again.
&EXIT
&COMMENT               Verify that the person calling the delete procedure
&COMMENT               is the same person that created the item.
$FIVE
&SAVE SENDER <CREATOR <VAR ITEMNO>>
&FORWARD FOUR <STREQUAL <VAR SENDER> <QUOTE <USERNAME> / <NODE>>>
&PRINT Sorry, you can only delete an item which you created, the item you
&PRINT specified is one for which you are not the 'creator'.  <chr 7>
&EXIT
```

CONTINUED NEXT PAGE

CONTINUED FROM PREVIOUS PAGE

```
&COMMENT                Sends message to all N. Boards requesting delete
$FOUR
&PRINT
&PRINT -----------------------------------------------------------------
&PRINT Your Notice will be deleted from ALL Notice Boards.  Each Board
&PRINT will send notification as it's able to delete your Notice.  BUT,
&PRINT there will be NO provision to save a copy of it.  If needed, you
&PRINT can copy it to your WorkArea upon completion of this command.
&PRINT -----------------------------------------------------------------
&PRINT
COPY <VAR ITEMNO> TO CLIP
OPEN "Using this N.Board"
SEND "Delete ITem#" TO "ALL BOARDS" OF "PUBLIC DISTRIBUTION"
&PRINT
&PRINT   The information shown on your screen above pertains to
&PRINT   copying done to send messages to each Notice Board, for
&PRINT   requesting that your item be deleted.  Please IGNORE.
&PRINT   However, you will now be asked to respond to a question.
&PRINT
&PRINT   ========================================================
&PRINT   Press the 'return' key to bypass the following question:  <CHR 7>
&PRINT
COPY FROM CLIP
DELETE 2
MAIL
CLOSE
&PRINT
&PRINT ---------------------------------------------------------------------
&PRINT The specified item will be removed from ALL Notice Boards within 75
&PRINT minutes.  Please WAIT !!    If you need to retain a copy, enter the
&PRINT following command immediately:                    <CHR 7>
&PRINT ----------------------------   COPY <VAR ITEMNO> TO WORKAREA   --------
&PRINT
&EXIT
```

Data Dictionaries: Bane or Boon

Stephen M. Butler
PROBUS International, Inc.
8815 - 106th St. E.
Puyallup, WA 98373

Introduction

Data Dictionaries (DD) have been around for over a decade. Even so, questions still arise: What is a DD? What are the benefits? Are these benefits real? The answers are varied and complex. The bottom line rests with the particular user and the application. This paper will point to the anticipated bottom line and give the circumstances under which the benefits should be realized.

DDs (Data Dictionaries) are the repository of data about data -- in short, meta-data. In similar fashion a Payroll system is the repository of data about employees and Accounts Payable the repository of vendors who need to be paid. Just as the Payroll and Accounts Payable data is important to the future development of a company, so is its meta-data.

The DD provides the mechanism to ensure that the benefits of database systems are fully realized. Just as a database reduces or eliminates redundancy of data (vendor name in six separate files) so the DD eliminates multiple definitions of the same item or, worse yet, distinct pieces of data defined as the same.

For example, in 1987 a major company in the Pacific Northwest was installing a commercial package on one of it's systems. This package made use of several databases. Ad hoc reports were needed that could quickly be developed using a report writer. The first step involved loading the database definitions into a DD. During this step several conflicting definitions for data names were discovered. In some places the Item Number was 6 characters while others had it as 10 characters. Worse yet was the discovery that these were indeed unrelated pieces of data. Different definitions were given the same name.

Consistency of data definition is very crucial for the successful implementation of 4th G/L and Relational DB technologies. It is also paramount to the survival of a company in this information age.

In fact, DD along with the DBA (Data Base Administrator) were projected to be the premier organization of the '90s. This combination was to provide the company with the knowledge of what information the company had available. With this information the company could chart its way through the application backlog using the leading edge of technology. The battle cry could have been "Corporate VP or BUST"!

BOON CITY

Within the HP-3000 community the early 1980s saw many vendors head toward the mecca of DD. Even Hewlett-Packard bought one and named it, appropriately, DICTIONARY/3000. The band wagon appeared all set to move. The key word became PROTO-TYPING. The tools were 4TH G/L and, of course, DD.

The benefit list became long (and sometimes emotional):

- common data definitions
- central repository of meta-data
- run time data resolution
- proto-typing
- automated documentation
- user views (ie, logical data design vs physical structure)
- language integration
- report writer generation
- data base definition
- etc.

If the above list leaves the impression that 4th G/Ls were the issue, than one understands where DD stood in terms of understanding. So, what were/are the benefits of, say, DICTIONARY/3000.

-- IMAGE Schema Generations

A project within Weyerhaeuser company kept the IMAGE schemas inside the DD. This was much easier than keeping separate schema files up to date. Modifications rarely caused problems (such as syntax errors or unreferenced data sets).

-- Ad Hoc Report Generators

A by product of this came in the form of INFORM as an ad hoc report generator. The data definitions were already in the DD. Some collections of data were brought together in INFORM GROUPS. A training program was instituted for the end users (several sites). This program stressed hands on examples using copies of live data. The trainees became familiar with both DICTIONARY/3000 and INFORM.

A recent conversation with the project manager revealed that this combination was very successful and still (some three years later) viable. In fact, this allowed the development staff to address some new major areas that would have been impossible otherwise.

-- Data Documentation

Kitsap County provides the development staff with updated element definitions. In fact, the development staff is expected to add new and/or modify existing definitions with the DD. Several elements exist that have nothing to do with IMAGE databases. These provide linkages for the development staff to locate the appropriate technical information.

-- On-line User Help

In addition, they developed an interface between VIEW/3000 and DICTIONARY/3000 that provides on line help to application users. The field names are entered into the DD along with the help text for the user. This allows them to produce a users manual that is viable across applications. The individual user departments are free to incorporate any specific policies or procedures into the manual. Thus the development staff is spared the burden of providing a totally customized manual. (The "help text" in the DD serves as internal documentation also. A nice double benefit.)

-- Language Support

Several languages now interface with various DDs to provide increased productivity. The authors primary experience has been with the TRANSACT language with DICTIONARY/3000. Various clients have reported program development gains of 8 to 10 times over COBOL. The author prefers a solid conservative 3 to 5 fold gain. Other developers with differing packages (PROTOS, COGNOS, SPEEDWARE, etc.) indicate similar advantages.

Almost every productivity aide in this arena has recognized the need for DD support. It is analogous to writing a single WORKING-STORAGE SECTION for COBOL and using it in all future programs with the compiler removing those portions that are unused.

All of the above made unique uses of the DD. None of them exploited the DDs to their full capability. This author wrote "DICTIONARY/3000 WALK THROUGH" to demystify the inner workings so that greater exploitation would occur. Then followed "DICTIONARY/3000: HUB OF SYSTEMS DEVELOPMENT AND DOCUMENTATION". Suffice to say, the tools necessary to fully utilize the ideas promulgated have not been forthcoming.

In fact, many sites still wonder what possible use they could have for a DD. Those who have found uses have also found frustrations.

-- Data Black Hole

Some sites have been chagrined to find their carefully loaded descriptions do not appear on the expected reports. For example, the DICTIONARY/3000 report of elements and sets within a database prints the ELEMENT to FILE description. The carefully entered ELEMENT descriptions are not available in that format. Conversely, the ELEMENT to FILE descriptions are not available with the ELEMENT listing.

The later makes some sense, but the former begs for an explanation. So far, no one has supplied one that is satisfying.

-- Competing Usage

As stated above, many vendors responded to the siren call and produced their own version of a DD. The major "productivity" language developers have found that a DD is needed. But, each has incompatible functions when compared to others.

The shop that has more than one such aide with the attendant discrepancies has relegated DD to the language support role. This is unfortunate as the major benefits of a DD are lost. Worse yet is the growing perception that a DD is something to be endured while attempting to gain productivity.

Some DDs provide for interfaces into DICTIONARY/3000 (and maybe other products) but the effort is time consuming and frustrating. Attempts to keep the DDs in sync are worse than handling redundant data in pre-database applications.

-- Ignorance is bliss

Any perceived benefits are lost in a shop using standard languages such as COBOL. These ignore the gold mine (some term it a coal shaft) of data definitions and require the development staff to re-type much of the structure. The available utilities do not provide the configurability needed to support variations between shops.

There are few pre-processors that retrieve data structures from the DD and insert them into a standard language. Those that do require their own specialized DD. This only adds to the growing burden of keeping everything in sync.

-- Time Warp

Perhaps it is better said "user warped"! With the available tools that directly change the database structure, not one will keep a DD in step. As time marches on, the difference between reality and the DD drifts from a hairline crack into a full concussion. The resulting headache is usually fatal for the DD and may even cast a coma over the associated shop.

-- Frozen meta-data

Some shops have found that the DD has provided such help as on-line user manuals. This close integration with the application is greatly appreciated until a change is needed in the on-line help. There is a story of a complete system recompile just to incorporate a grammatical change in the help text.

Even more amazing is the concept of compiled DD. Perhaps the first such encounter is with the IMAGE root file. For obvious reasons, this does need "compilation".

Conversely, the DD is a central library of information that is dynamic. Even as applications and systems change so must the information library. It is not true that changes in the information library (DD) should require change in the application.

-- Friday on strike

The utilities that do exist tend to support the incorporation of information into the DD. This information is usually in a raw form that needs further clarification by the development staff (such as where decimal points really belong in an IMAGE J2 field). Very few utilities travel the opposite path.

The development staff is further stymied by the lack of support during the migration from development to production. This is especially true where there is a phased release going to more than one site. The above comments regarding interaction between DD of competing vendors could be repeated here. The usual method of transfer results in production seeing everything development had in the DD at release time. The concept of phased releases is foreign to most DDs.

Even further astray is support for distributed DDs. Many might claim that the current multi-vendor solution is distributed. The lack of support within a single vendor's DD makes this situation nearly identical to the multi-vendor bottleneck.

OUT OF THE QUAGMIRE

The above sounds hopeless. While the situation is more dismal than expected, it certainly isn't hopeless. The route from Bane Villa to Boon City need not end in the Quagmire. The mile posts along the way must contain the following.

-- More Useful Utilities

The Data Black Hole concept needs reversed. Documentation and other descriptions that are input must be readily regurgitated by the DD. Such areas of major concern are User On-line Help and User Manuals. The greatest distress for the development staff rests with end user documentation. Any utility that will reduce this burden (notice the word REDUCE) will be a great benefit.

Many shops take the data descriptions from the DD to a flat file and then proceed to rework them into user documentation. An utility that understands user documentation and is configurable via a meta-language in the DD will greatly facilitate in this area.

A similar utility should allow the on-line user direct access to the documentation stored in the DD. Such items as Departmental Policy and Procedures, Screen Level Help, and User Enterable Field documentation could be readily available at a keystroke.

Data validation rules would also be accessible. The human readable version could be generated as needed from an internal control language. This language would allow the development staff to incorporate data validation codes within the application.

In fact, the DD should provide language snippets to go in COPYLIBs or otherwise be included in standard 3rd generation languages (COBOL, FORTRAN, ...). These snippets would be both Working-Storage (COBOL) and procedural code.

One major area of the DD is the documentation of what programs use which elements from identified files (datasets, flat files, etc.). The natural outgrowth is code written in any language to retrieve and/or update said elements. This need not be a complete program but would be code the programmer could slap into a program and utilize. Quick integration into a program is the key ingredient.

Another key ingredient is internal documentation that the DP auditors love so much. Since the DD already knows the structure of the databases, it should provide the standard chart consisting of triangles and trapezoids. It should also be able to reduce the number of line crossings to a minimum or even eliminate them with unique but reasonable layouts.

Similar utilities would provide program level documentation. Something akin to HIPO charts or high level data flow diagrams should be available.

-- Productivity and Development Aides

There are many areas in which development can be aided. The standard 4th G/L approach has been in the arena of program construction. Other areas are just as critical to the complete life cycle of an application. The initial design phases are usually completely redeveloped in later stages. This initial effort needs to be captured in a manner that is easy to regurgitate in the form needed later.

In fact, by the time detailed system design is started there is a great deal of information known that should be in the DD to facilitate the data structure design. With the added knowledge of data dependencies the DD should provide a normalized (at least 3rd normal form) logical data design. Given a few rules regarding performance considerations, the system could even produce the first cut at a physical database layout. (In the HP-3000 world the distinction between logical and physical design is greatly blurred.)

-- Extendible and Configurable

While the current state of DD art is very far from what has been described so far, it is doubtful that any comprehensive DD solution will be the answer to every need. Therefore the basic structure needs to be extendible by individual sites. The basic implementation should be very rich to negate many needs for the extendible. But, the DD should support a very extensive capability for extension.

The basic structure should be amenable to some configurability. Such things as the ELEMENT NAME size along with other attributes need to be configured by the individual sites.

-- Vendor Integration

The current plethora of vendor specific DD systems is a major roadblock. The DD of the future needs to support varied needs according to other vendor software. In fact, the ideal DD would allow the installation procedures to compare the current DD structure against the needs of the new package. Here again, Extendibility and Configurability come to play.

This is akin to the varied remote control devices for TVs, radios, video decks and satellite dishes now on the market. The solution is one remote control that can learn all the functions of the others combined. The DD of the future must be able to incorporate the needs of other packages.

-- Language Support

Mention has already been made of utilities that would assist the programmer by generation of data structure and procedural code. A further integration is needed whereby the compilers of 3rd G/L software can interrogate the DD to supply field definitions or even round out some sections of procedural code.

This sounds like infringement on the current slate of 4th G/L generators. However, most of these generators are lost in the foursome of ADD, UPDATE, DELETE and REPORT functionality. They do not address the more complex issues of A/R aging, Payroll processing, or General Ledger cycles.

The 4th G/L generators will become capable of more advanced processing. Likewise the DD should become smarter about languages and how to generate code for them. Even so, it is incumbent upon the 3rd G/L compilers to learn about the DD.

-- Programmatic Interfaces

All of the above areas demand smart programmatic interfaces into the DD. These interfaces must be bi-directional. In addition, the inquiring software must be able to discern what extension are in place and how the current site has configured the DD.

This would allow a new package to automatically extend the DD where needed and modify its own code to comply with local configurations where appropriate. Naturally, there will always be areas where the shop will have to choose between alternatives. The choices will be easier when the software lays out the pros and cons.

Much current software could benefit by direct interrogation of the DD. Major examples are the ADAGER and DBGENERAL utilities. Ideally the DD would drive either or both utilities (configurable option!) to change the database from what was to what is to be (or not to be).

Alternatively, either or both of these products should be able to update the DD to reflect the actual changes made.

TO EPCOT

The reader may conclude that the author has gone completely futuristic. This may be even more apparent when a careful review indicates the need for an EXPERT SYSTEM. Indeed, the Data Dictionary Software must become an EXPERT SYSTEM. It needs to be an expert regarding systems development, design and documentation.

Various interfaces must be provided including the board room interface where questions such as, "If we collect this addition information regarding our clients (or products or ...), will we have a competitive edge in our industry?" Therefore, this system must also become an expert regarding the company and industry it is functioning within.

The programmer interface is much different. Much of the grunt work of coding would be done by the DD system. That would leave the programmer free to concentrate on the more unusual requirements. These requirements would be researched through interaction with the DD to gather available documentation. Missing information would be incorporated by the programmer in accordance with the local shop style.

The user interface becomes subtle. A pause, erroneous input or other indications of confusion would be met by a system ready to analyze the current situation and suggest alternatives. Always would be the fall back where the user would interrogate the DD for information. This may be more detailed, more general, or further afield than initially offered.

To do this, the DD system must expand its own knowledge base. That is the major premise of expert systems. They learn from the information that is incorporated therein. This implies that the DD would learn about changes made to itself and incorporate that knowledge into its expert base.

Not only would this expert system have human interfaces but it would allow inquiries from applications and other systems. This would range from the current set of edit specs for a field to the current configuration of the DD.

CONCLUSION

Bane Villa or Boon City? The current situation is both. The maintenance of element definitions and IMAGE schema files is a snap. Full DICTIONARY/3000 implementation suffers from the Black Hole theory. Multiple vendor shops with the attendant DDs have found the dream of consistency across dictionaries to be elusive.

So what is a shop to do? The simple things:
- Keep the IMAGE structure in the DD.
- Maintain the Element documentation both for technical level and end user help facilities.
- Utilize the ad hoc report generator interfaces.
- Where applicable, utilize the 4th G/L generators.

And, finally, let your vendors know that integration is the future. The "divide and conquer" situation must become "let's hang together or we'll hang separately."

Interestingly, help may be coming from an unexpected source. The AI expert system developers have found relational technology to merge very nicely with knowledge basis. Their solutions for managing this information base are starting to sound like Data Dictionary systems. Soon someone will make the giant leap that intertwines the expert system with the description of its knowledge base. A smart DD is nothing more than an expert system driving the description of itself in order to facilitate its own development with the goal being design, documentation and development of expert applications.

Building a Home for Your HP 3000's
Data Center Design, Construction, and Operations

Harry J. Krommer
Procter and Gamble
Cincinnati, Ohio 45201

Introduction

Electronic computer/data processing equipment has become a vital and commonplace tool for business, industry, government, and research groups in recent years. The use of such equipment is a direct result of the technological breakthroughs which have made the equipment available and the increased complexity of modern business, industrial, governmental, and research needs. Particularly pertinent are the increasing number of variables which must be taken into consideration in everyday decisions - overlooking any one item may spell the difference between profit and loss, success or failure, life or death. To keep track of all these variables, the computer offers practical answers.

This equipment is being used on an ever increasing basis to process large amounts of statistical, problematical, or experimental information, and to print out or display answers or information in very short periods of time. More and more reliance is being placed on the equipment to perform the repetitive, the experimental, and in some cases, even the whole programming operation for business, industry, government, and research groups.

This increased reliance on the computer has pushed the need for the ever more reliable computer and thus the necessity of "taking care" of the computer. While a computer's "needs" can be very exacting, they are few in number. Provide a computer with controlled temperature and humidity, dust-free air, freedom from radio frequency interference, and an uninterrupted source of filtered, regulated electric power and you'll keep those costly upsets to a minimum. Adequate fire protection, provisions for water-leak detection, and a realistic facility security system will defend against three more sources of trouble. An environmental-conditions monitor can put the finishing touches on a complete environmental-control package that will keep your computer on-line with a minimum of fuss. Each of these areas merits closer attention.

GENERAL DESIGN/CONSTRUCTION ISSUES

- Is the site large enough to contain the equipment? You should allow approximately 300 sq. ft. for each average size series 70. It is also prudent to plan at this time for expansion of the data center. If a second machine or significant expansion (i/o bay, disk drives, etc.) of the existing machine is likely then doing the construction now to include these needs would greatly reduce the impact on the future operations. If expansion is a probability at sometime in the distant future, you still need to insure that space adjacent to the existing room can be secured when the needs become current.

- Will the site/construction provide physical security? It was "fashionable" at one time for a company to put their computer operation on display. This showed their customers and visitors that they were an "automated modern company". Today, computer rooms are tucked away and do not have windows for viewing the operation. The walls are usually constructed of concrete block that go from floor to the primary ceiling. The doors to the room are steel fire rated doors with some form of locking to provide entry only to the authorized personnel. There are no signs in the rest of the building indicating the location of the data center.

- Has the site been checked for unique internal or external conditions such as excessive dust, vibrations, magnetic or electrical fields? The site selection also needs to be evaluated on this criteria that can be much more difficult to "see". Is the site near a transmitting tower, large motors, or motor controllers. These RF problems can be among the most elusive problems to diagnose and costly to solve. If in doubt an RF study by a consulting firm is in order.

- Can you get the equipment to the site? Walk the path that the equipment will take from the receiving dock to the room. Are the doorways big enough? Is the elevator large enough and can it handle the weight? Are the hallways wide enough and is there room to make the required turns with the equipment? Is the floor strong enough to support the equipment plus the equipment being used to move it?

- Are you meeting all code requirements? There are national, state, and local codes that must be met for a computer room. If you do not know these code requirements the best way to insure your compliance is to have knowledgeable people as your resources (from inside or outside of your company).

- Is raised floor necessary? Raised floor provides for very flexible cabling needs, gives a neat appearance, and safer for the personnel. It also provides the best method for distributing the conditioned air to the needed locations in the room. Raised floor should be a minimum of 12 inches above the primary floor if possible. Heights down to 6 inches require that the cabling under the floor be more managed so as not to restrict the air flow. Raised floor adds approximately $10/sq. ft. to the room cost, so the cost/benefits of it are well worth the added cost.

- Emergency stop buttons are required by the national electric code to be located at the exit doors of the computer room. These buttons should be shielded from accidental pushing and must turn off all power in the room except for the lighting.

- Emergency lighting should be provided in the room. You can not imagine what dark is until you are in the back of the computer room with no windows and the lights go out.

- An annunciator from the building fire alarm should be placed inside of the computer room. The rushing air sounds inside of the room make it impossible to hear anything that is located outside of the room.

- A high temperature cut-off switch should be connected to the computer grade power. In the event that the room reaches 85 degrees F, the power to the computer should be cut-off to automatically force it down. The system manager who allows the computer to be "cooked" will have a miserable life for the next 6-12 months while stressed components fail one after another.

- All computer grade power should be kept under the raised floor. The electrical outlets on the wall should be standard building power. This prevents the maintenance person from plugging in a piece of equipment that will generate noise on the output side of the clean power system.

Now let's look at three of the more critical components in maintaining a good computer environment.

ELECTRICAL POWER

ELECTRICAL POWER CONCERNS

Sensitive computer equipment requires special care. In today's competitive business environment, there is no question...computers play a major role in the survival of almost every company. Mainframes, minis and personal computers support entire operations from purchasing and production scheduling to word processing and accounting functions. Unscheduled downtime is frustrating as well as expensive. One of the leading causes of computer downtime stems from problems associated with the electrical power.

Recent studies show the typical computer site is threatened approximately 128 times a month with some sort of power disturbance. Obviously the types of power problems, severity and number of occurrences will vary from site to site. Experts say power disturbances are responsible for 62% of all computer problems.

Overall, our nation's power is 99% reliable. It's the quality of the power that has become the focus of increasing attention. In the past consumers have evaluated power quality by the simple standards of present (available) or not present (not available). With the implementation of micro circuitry in electrical devices such as computers, the requirement for more perfect power has increased. Over the years our standards have become much more sophisticated.

Not all power disturbances originate on the utility side of the meter. Consumers can generate problems simply by the normal use of electricity. Everyday activities like the operation or switching of a large load within your facility can generate a line disturbance. An example is heavy air conditioning loads operating during the summer months which can depress capacities and cause the voltage to drop. During this low voltage or sag condition, power can momentarily fall below the tolerance limit of the computer, long enough to crash the system or cause severe hardware damage. Another instance of "power pollution" is poor grounding. Multiple equipment grounds that are incorrectly isolated can cause ground loop currents that put noise into signal circuits. These are the reasons that every computer site is subject to power disturbances.

Unfortunately for computer users the list of power disturbances and the origins go on and on. The fact remains, a reliable source of quality electrical power is critical for dependable computer operation. The fact remains, a reliable source of quality electrical power is critical for dependable computer operation. Considering the significant investment made in automation, the computer and its power deserve special care. Remember, it is not only the equipment you want to protect but the valuable information stored inside as well.

POWER DISTURBANCES AND DIRTY POWER

Comprehensive studies have been done on power disturbances, typical causes, and the affects on computer systems. The experts have classified disturbances into three categories: Type I, Type II and Type III.

Power disturbances can happen at any time, and can be caused by events such as lightning strikes, utility equipment failures, large load changes, arc welders, motor speed controllers and other randomly occurring events.

Figure 1 thru 3 summarize each type of disturbance and depict the affects of each on a normal electrical (AC) sinewave.

PINPOINTING POWER PROBLEMS

System crashes, processing errors and repeated maintenance calls for computer repairs are obvious indicators of power problems. You don't have to live through these crises to find out there is a problem with power. The most effective way to pinpoint potential problems is to call in an expert for a power survey.

A power survey is a complete evaluation of the incoming power as well as the electrical design of the facility. A line disturbance analyzer is connected at the power input to measure and record the incoming power, the power actually used by the computer. Usually the analyzer is left in place long enough to collect data typical of the site. The electrical design is also evaluated in terms of circuits dedicated for the computer and equipment grounding.

This on-site study provides the data to identify specific power problems. It also provides a basis to determine the need or level of protection or refinement for the computer and its power.

Today computer vendors may recommend the installation of buffering or corrective devices to protect against power disturbances. The necessity of minimizing or totally eliminating disturbances depends on the sensitivity of the computer operation and the individual operating requirements. The question is not one of choosing protection but rather getting the return you anticipated on your computer investment.

MINIMIZING OR ELIMINATING THE EFFECTS OF POWER DISTURBANCES

With recent advancements in computer room power protection, most power disturbances can be minimized and even completely eliminated. In some cases it may only be necessary to modify the electrical design of the facility to correct a disturbance. Other cases require the installation of power conditioning or power synthesizing equipment to achieve optimum results.

In the latter case there is a number of devices to choose from, each offering a different level of protection.

POWER CONDITIONING EQUIPMENT

Power conditioning equipment is effective on Types I and II disturbances. These devices enhance the power and correct very specific problems. Conditioning devices include: 1. Isolation Transformers 2. Spike Suppressers 3. Voltage Regulators These are inexpensive solutions that provide a minimal level of protection. Often combinations of these devices must be used in order to obtain desired results.

POWER SYNTHESIZERS

Power Synthesizers actually use the incoming utility power as an energy source to create a new sinewave free from any disturbances. These regulatory devices can be as much as 99% effective against power disturbances. Synthesizers include: 1. Magnetic Synthesizers with the capability of generating a sinewave with the identical frequency (60 Hertz) of the incoming power. Since these devices use the utility power as an energy source to rebuild the sinewave and they are effective only against Types I and II disturbances.

2. Motor Generators use an electric motor to drive a generator which provides electrical power. If equipped with a heavy fly wheel, the MG may ride through momentary outages. These devices are relatively costly in terms of the initial purchase price and maintenance. MGs are effective against Types I and II disturbances.

3. Uninterruptible Power Supplies or UPSes provide the best protection against all types of disturbances. The system consists of a rectifier, battery and inverter working together to provide a continuous flow of power to the computer even when the utility power is completely off. The amount of protection time is determined by the battery capacity you choose. Generally sufficient battery capacity is chosen to conduct an orderly shut down or to start up a diesel generator. Two types of UPSes are available, stand-by and on-line. Stand-by are the least expensive of the two technologies but are somewhat limited by their inverter. With the stand-by technology, the switchover time could be long enough to lose the critical computer load. Any problems with the inverter cannot be detected until the crucial moment when it is required to carry the load. An on-line UPS differs in that the inverter is on at all times. There is no switchover time involved to make the transition to the battery discharging mode. An on-line UPS supplies the computer with continuous conditioned and regulated power. Because this type of system is on-line continually, potential problems can be sensed in real-time rather than waiting for a power outage to find out whether

all components are operational. A UPS is the ultimate in power protection and generally the most expensive.

CONCLUSION

The cost of each solution increases with the level of protection desired. The more protection required...the greater the investment.

Using a cost analysis, the true value of power protection is realized when compared with the cost of downtime and hardware damage from a single disturbance.

AIR CONDITIONING

Computers, like giant electric furnaces, generate vast amounts of heat and their components are very sensitive to extremes of temperature, humidity, and the presence of dust.

In the past, a substantial percentage of computer installations were designed to utilize central refrigeration systems with either a built-up or a modular fan-coil air treating distribution system. The pendulum has now swung in favor of unitary expansion systems.

Today, it is estimated that from 65% to 75% of the environmental control systems being designed for new or remodeled computer rooms employ unitary systems.

Why this significant trend away from central systems? Three principle benefits which accrue to the owner-user are: simplicity, flexibility, and economy. In conjunction, system security is becoming more of a factor in the application of unitary systems. If the security of the computers and their peripheral equipment is important-and it is-then why not make every effort to maintain the same degree of security over the environmental control system? Physically locating the entire air conditioning system within the computer room itself, with simple connections to indoor or outdoor heat rejection systems-which can be located in any secured area of the building system, virtually assures total systems security.

PRECISION AIR CONDITIONING VS. COMFORT AIR CONDITIONING

Precision air conditioning is the simultaneous control of temperature, humidity, air motion and cleanliness in a specified area, continuously and accurately. It is as far in advance of comfort air conditioning as the computer is of the adding machine.

Modern office buildings have long accepted comfort air conditioning as a definite influence on increased productivity,health, attendance, and comfort of their tenants. The design of the system varies with the types of buildings and their geographical location.

Industrial air conditioning is very common where certain processes require control of temperature and humidity not only for the sake of the product process, but also the factory worker. Generally speaking, the control for these types of systems is not accurate enough for today's sophisticated electronic equipment such as in computer rooms. The computer room requires precision air conditioning year round.

Conventional comfort air conditioning is not precision air conditioning. Conventional systems cannot handle computer room loads that demand close tolerances in both temperature and humidity year round.

Comfort air conditioning primarily conditions people. Precision air conditioning primarily conditions equipment. There are distinct and important differences in the design parameters.

1. DESIGN CONDITIONS The average skin temperature of people is 80 degrees F. If the room is cooler, bodies radiate heat; if warmer they absorb heat. Consequently, comfort air conditioning systems are designed at 80 degrees F in the summer, and at 60 degrees F in the winter. This is too wide of a range for computer room applications. A computer radiates considerable heat and requires a stable temperature of 72 degrees F to 75 degrees F.

2. SENSIBLE HEAT RATIO (See Figure 4) People emit latent heat through normal metabolism, and the heat contains moisture. Equipment gives off dry electronic sensible heat which is moisture-free.

The sensible heat ratio - the amount of sensible cooling as a percent of total cooling capacity (sensible plus latent heat) - is 65% to 70% of the total load in a comfort system. The dry heat generated by computers requires a sensible heat ratio of 90% to 95% from the cooling equipment. This creates additional confusion when the only measure of cooling capacity is "tons". A comfort air conditioning system nominally rated at 10 tons may only provide 6 or 7 tons of precision air conditioning. This has resulted in unpleasant circumstances when the room cannot be cooled; there may not be enough cooling. This same concept holds true with smaller 1-3 ton requirements.

3. LOAD DENSITY (See Figure 5) Because of this high

sensible heat, the cooling capacity in terms of BTUH per square feet is greatly increased. Office buildings, where the total load results from many people and considerable outside air, are designed for 40 to 48 BTUH per square foot. Data centers average 120 to 240 BTUH per square foot, 3 to 5 times as much as in a comfort system.

Translating this to another "rule of thumb", square feet per ton, we have:
COMFORT AIR CONDITIONING-250 to 300 sq.ft./ton.
DATA CENTERS-50 to 100 sq.ft./ton.

4. AIR QUANTITY Comfort systems with room cooling design temperatures of approximately 80 degrees F normally supply 350 to 400 CFM per ton of cooling capacity. Computer rooms, because of the lower design temperature requirement of 72 degrees F, require 50% to 60% more air; on the order of 500-600 CFM/ton. In addition to air quantity, the air distribution pattern is critical.

5. HUMIDITY CONTROL Both humidity and temperature must be "just right" for computer rooms or there can be expensive shut-downs. Comfort air conditioning systems do not normally have any humidification capability. Dehumidification occurs during cooling modes of operation, but will not occur if the humidity level rises and with no increase in temperature. Precision air conditioning systems provide for the simultaneous control of humidity and temperature. They also insure that the humidification and dehumidification operate separate from each other.

6. ANNUAL HOURS OF OPERATION Usually comfort air conditioning is in operation some eight hours a day, five days a week, from April to September. That's almost 1200 hours intermittently. Precision air conditioning operates continuously all day, every day, all year long. That's 8760 hours, non-stop.

7. PRECISION CONTROLS Computer room air conditioning demands temperature and humidity controls which are fast acting, and capable of holding rood limits within a temperature swing of plus or minus 1 degree to 3 degrees F and a humidity swing of plus or minus 2% to 4% RH. Without fast-acting precise control, computer hardware and peripheral equipment are susceptible to many types of shutdown problems.

COMPUTER ROOM PROBLEMS Problems such as equipment failure ... hot spots ... low air quantity ... condensation ... tape deterioration ... processing gibberish ... read-write errors ... paper sticking ... card jams and head crash are the result of the improperly controlled temperature-humidity and cleanliness due to inadequately designed computer room air conditioning.

INVESTMENT Computer systems represent a substantial investment of tens of thousands of dollars...often hundreds of dollars per hour! Heat, moisture, and dirt can cause breakdowns resulting in tremendous financial losses. To protect against these losses, a unitary air conditioning system is relatively inexpensive.

Investment in precision air conditioning can be as low as 1% of the value of the computer equipment installed in the data processing center.

SYSTEM OPTIONS There are four types of computer room environmental systems...air cooled...water cooled...glycol cooled...chilled water.

UNITARY

Air Cooled is the simplest type. It uses refrigerant to absorb the computer room heat. Then, in a closed loop, it takes the heat outside through an air cooled condenser where it is discharged into space.

The Water Cooled system uses refrigerant to absorb heat but it transfers the heat to a water cooled condenser that is built into the unit. The cooling water is circulated through an outside cooling tower where the heat is removed.

The Glycol Cooled system also uses refrigerant to absorb the heat but transfers it to a glycol solution through a condenser within the unit. The solution is then pumped to a drycooler which dissipates the heat to the outside air.

CENTRAL

In its simplest form the Chilled Water system uses water to directly absorb the heat in a computer room. This is usually done by a chiller which removes the heat from the water and returns chilled water back to the unit. This is not a totally independent system unless a separate chiller is provided for the computer room.

ENERGY EFFICIENCY RATIO AND OPERATING COST

Not surprisingly, EER data and operating cost analyses favor the simplicity of direct heat rejection from refrigerant to outside air-as in the case with air cooled systems.

Acknowledging key features in basic equipment design which include the use of semi-hermetic compressors, shallow evaporator coils coupled with low face velocity for high sensible heat factor performance and low fan horsepower-EER's on the order 8.5 to 10.5 are not uncommon for air cooled unitary systems. And these compare to 6.5 to 7.0 for comparable capacity glycol systems utilizing sealed hermetic compressors.

Recent computer studies for industrial users and also experience with the Federal Government (GSA) Value Management Program, point to a significant increase in selection of air cooled unitary systems.

An on-line system typically would require equipment operation 8760 hours per year. Experience dictates approximate operating time for various possible modes of system operation will be:

Percent	Mode
50	Full cooling
20	Dehumidification w/full reheat
20	Dehumidification w/no reheat
5	Heating and humidifying
5	Full cooling and humidifying

For a given system, once the KW required for each of the above mentioned operating modes is determined, it is a simple matter to apply the time percentages, calculate the total KWH per hour of operation - and project annual operating cost over an 8760 hour year at an average cost of power of so much per KWH.

SUMMARY The user is encouraged to review the basic differences between comfort air conditioning systems - and precision air conditioning systems as commonly applied to computer rooms today. Trends away from central systems to unitary systems for reasons of simplicity, flexibility, economy and security should also be noted. The selection of air cooled unitary systems for the primary reasons of simplicity and economy represent a significant trend in the industry.

FIRE PROTECTION

Much has been written on the procedural steps required for study before installing electronic computer/data processing equipment. These requirements embrace selection of proper equipment, checking and planning for areas to receive the equipment, utility requirements, orientation and training of personnel to operate the equipment, as well as consideration for expansion of the initial facility. One other factor should be included in this vital study - namely, protection against fires of either accidental or deliberate origin.

In addition to the hazards of fires from accidental causes, many computers and data processing installations have become prime targets for sabotage and arson.

Oftentimes, the strategic importance placed upon electronic computer/data processing equipment by the user is vitally tied to uninterrupted operation of the system. Consequently, by the partial or entire loss of this equipment, an entire operation of vital nature could be temporarily paralyzed.

Not to be overlooked are the "one-of-a-kind" electronic computer systems. These are the "custom-made" models that are designed to perform specific tasks. Replacement units for this type of equipment are not available and the probability of the existence of duplicate facilities, which could be used to perform vital operations in the event the "one-of-a-kind" system is partially or totally impaired by a fire, is remote.

Computer equipment and materials for data recording and storage may incur damage when exposed to elevated sustained ambient temperatures. The degree of such damage would be variable depending upon exposure, equipment design and the composition of materials for data recording and storage. The following are guidelines reflecting concernable sustained ambient temperatures:

a) Damage to computer equipment may begin at a sustained ambient temperature of 175 degrees F with the degree of damage increasing with further elevation of the ambient temperature and exposure time. This temperature would be the temperature inside the computer cabinets that the components "see" and not the temperature of the computer room.

b) Damage to magnetic tapes, flexible discs, and similar materials may begin at sustained ambient temperatures above 100 degrees F.

c) Damage to hard disc may begin at sustained ambient temperature above 150 degrees F with the degree of damage increasing rapidly with further elevations of ambient temperature.

d) Damage to microfilm may begin at a sustained ambient temperature of 225 degrees F in the presence of high humidity or at 300 degrees in the absence of high humidity.

Planning for fire protection is vital due to an organization's dependence upon the computer equipment. Once management commits itself to a program of dependence on any such equipment, simple economics dictates doing away with former methods and procedures. The personnel, equipment, and facilities are no longer available to pick up the load assumed by the data processing equipment if it is put out of operation by fire or other unforeseen occurrences. Often, the major cost involved to management by disruption of the computer operation is from business interruption rather than from the actual monetary loss represented by the equipment itself.

Exposure to destruction can come from within a computer cabinet, from within the equipment room, from the immediate area around the data processing room, from floors above and below the computer, and from outside of the building in which the equipment is located. This exposure can be evaluated and then controlled as needed.

METHODS OF PROTECTION:

- halon or carbon dioxide flooding under raised floor - a high level, fast acting method in which halon vapor is discharged automatically into the space under the raised floor. Halon use minimizes the noxious, corrosive effects of smoke vapors produced from the combustion of PVC insulation on computer cabling. Smoke films which remain may cause unexpected and recurring operational problems months after a cable burns. Thus it is vital to quench the generation of smoke in under floor areas as quickly as possible. Carbon dioxide is a potential alternate to halon but the freezing effects and density of this gas can make it less effective in tight, under floor areas. Use this protection (in addition to primary room protection methods) where cables exist under a raised floor.

- local halon flood of enclosure panels - a fast acting protection method which involves directing halon nozzles directly into electronic equipment enclosures. This method reduces the cost of filling an entire room, but carries some risk that the fire won't be extinguished unless all sources of ignition are surrounded by halon vapor.

- carbon dioxide flooding of room - enough carbon dioxide gas is discharged into the room (upon sensing a fire) to suffocate combustion by displacing all oxygen. Restrict use to unoccupied rooms unless special alarm and delayed actuation precautions are taken. **Special Note** ... this method represents a severe personnel hazard and the required delayed actuation allows a fire to progress after detection.

- halon flooding of room - a high level, fast acting protection method in which halon is discharged into the room from fixed ceiling heads (usually discharge also occurs under raised floor from smaller heads placed in the under floor space). Halon expands quickly throughout the room. The halon/air mixture chemically breaks the combustion chain and the fire goes out. Maintaining the mixture for a fixed period of time to minimizes the risk of re-ignition. Use sprinklers as a back-up on critical equipment systems.

- auto wet-head sprinklers in room - disaster level protection which uses fusible links or quartzoid cylinder wet head sprinklers within the computer room. Each head is individually activated by exposure to ceiling temperatures of at least 165 degrees F. This protection senses catastrophe or disaster size fires, but may not respond to minor electrical short circuits or burn-outs.

- pre-act water sprinklers in room - disaster level protection which reduces the risk of accidental water release, and uses an early warning detection system to fill the normally empty sprinkler

- building or perimeter sprinklers external to room - external area protection which minimizes hazards external tot he computer room. It may protect the computer room from major fire damage should a fire start outside of the room. Only the outside perimeter area around the computer room may be sprinklered if the surrounding building area is not. Use this protection in addition to primary protection inside the room.

A CLOSER LOOK AT HALON

Halon 1301 is a colorless, odorless, electrically non-conductive gas with the chemical name "bromotrifluoromethane". It is used in a fire suppression system to protect valuable or irreplaceable materials and equipment which could be damaged or destroyed by other types of fire extinguishing agents. Halon 1301 extinguishes fire by inhibiting the chemical reaction of fuel and oxygen. It is a "clean" agent because it leaves no water, foam, or powder residue behind to damage delicate equipment.

Halon 1301 versus water

Water is a conductor of electricity and generally not selected for fire protection of electrical equipment. It also is not recommended for use on fire in record storage areas, libraries, rare document rooms, etc. because it often causes more damage than the fire itself.

Halon 1301, on the other hand, is highly effective on electrical equipment since it is a non-conductor of electricity. It leaves no residue so it won't harm records, books, and other valuable documents.

Halon 1301 versus carbon dioxide

Carbon dioxide will not conduct electricity nor will it leave residue which could damage equipment. But carbon dioxide works by displacing oxygen from the atmosphere of the fire area and this results in a dangerous situation for personnel.

Halon 1301 works by interfering with the combustion process, not by diluting or displacing oxygen. Accordingly, an atmosphere containing enough halon 1301 to put out the fire (5-7% concentration) also will safely support human life (check with a fire protection vendor for the safe exposure times for halon).

SUMMARY

In short, the computer has some basic needs that when fulfilled will insure you of maximizing the uptime of your investment. HP's "Site Preparation Set" provides an excellent guide toward this goal and the HP customer engineer is another excellent source.

Maintain and test your equipment per the manufacturers recommended P.M. schedule and it will support your needs for maximized computer uptime.

THANK YOU

My sincere thanks to Mr. Dick Hackman (HP Senior CE, Procter and Gamble Major Account Team) not only for his collaboration and support on this paper but also for the many years of dedicated service supporting the HP effort within P & G.

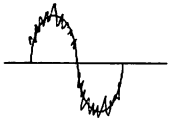
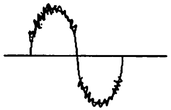
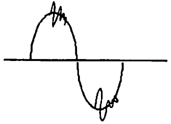
REFERENCES

"Site Preparation Set", Hewlitt Packard Company

"Answers to Four Critical Questions on Electrical Computer Power", Liebert Corporation

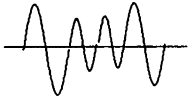
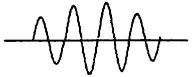
"The Seven Elements Every Manager Should Know About Computer Air Conditioning", Liebert Corporation

"Protection of Electronic Computer/Data Processing Equipment 1976", National Fire Protection Association Inc.

POWER DISTURBANCES	DEFINITIONS	CAUSES	POSSIBLE COMPUTER SYMPTOMS
COMMON MODE DISTURBANCES 	Impulses and EMI/RFI noise with respect to ground superimposed on the power conductors. Amplitude–Millivolts to several volts for noise. hundreds of volts for impulses	Radio transmission. Normal computer operation. Arcing contacts. Lightning. Poor grounding and shielding.	Incorrect data transfer from CPU to disk or tape. Terminal or printer errors. Input/output hardware damage. Power supply damage.
NORMAL MODE NOISE 	Low level signals superimposed on the power sinewave. Amplitude – 0.5 V to 25 V.	Normal computer operation. Switching power supplies. Power line modulation equipment (i.e. Simplex™ clocks). Motor speed controllers.	Processing errors. Incorrect data transfer from CPU to memory. Printer or terminal errors.
NORMAL MODE IMPULSES and RINGING TRANSIENTS 	Typically a narrow, fast-rise voltage variation. Can be followed by a damped oscillation decaying to nominal in less than one cycle. Amplitude – 50V to 6kV. Duration – 0.5 usec to 2000 usec.	Switching loads on or off. Normal computer operation. Fault clearing. Utility switching. lightning.	Incorrect data on disks or tapes. Processing errors. Printer or terminal errors. Hardware damage.

Type I

FIGURE 1

POWER DISTURBANCES	DEFINITIONS	CAUSES	POSSIBLE COMPUTER SYMPTOMS
SAGS 	A low-voltage condition on one or more phases. RMS voltages below 80-85% of nominal. Duration - Greater than 1 cycle.	Ground faults. Starting large loads. Inadequate power system capacity. Utility switching. Utility equipment failure. Lightning.	Computer system crashes. Hardware damage.
SURGES 	A high-voltage condition on one or more phases. Voltages above 110% of nominal. Duration-Greater than 1 cycle.	Rapid load reduction. Utility switching.	Hardware damage.

Type II

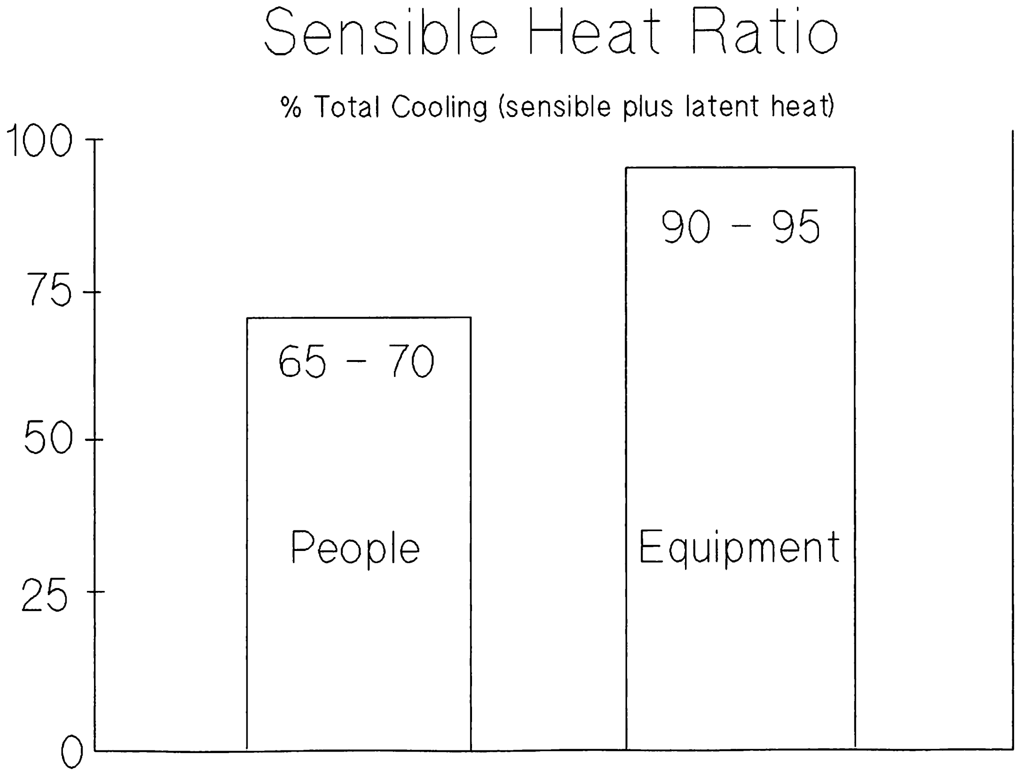
FIGURE 2

POWER DISTURBANCES	DEFINITIONS	CAUSES	POSSIBLE COMPUTER SYMPTOMS
OUTAGE 	A zero-volt condition lasting longer than a half-cycle.	Ground faults. Equipment failure. Accidents. Utility Equipment failure. Lightning. Acts of nature.	System crashes. Hardware damage.

Type III

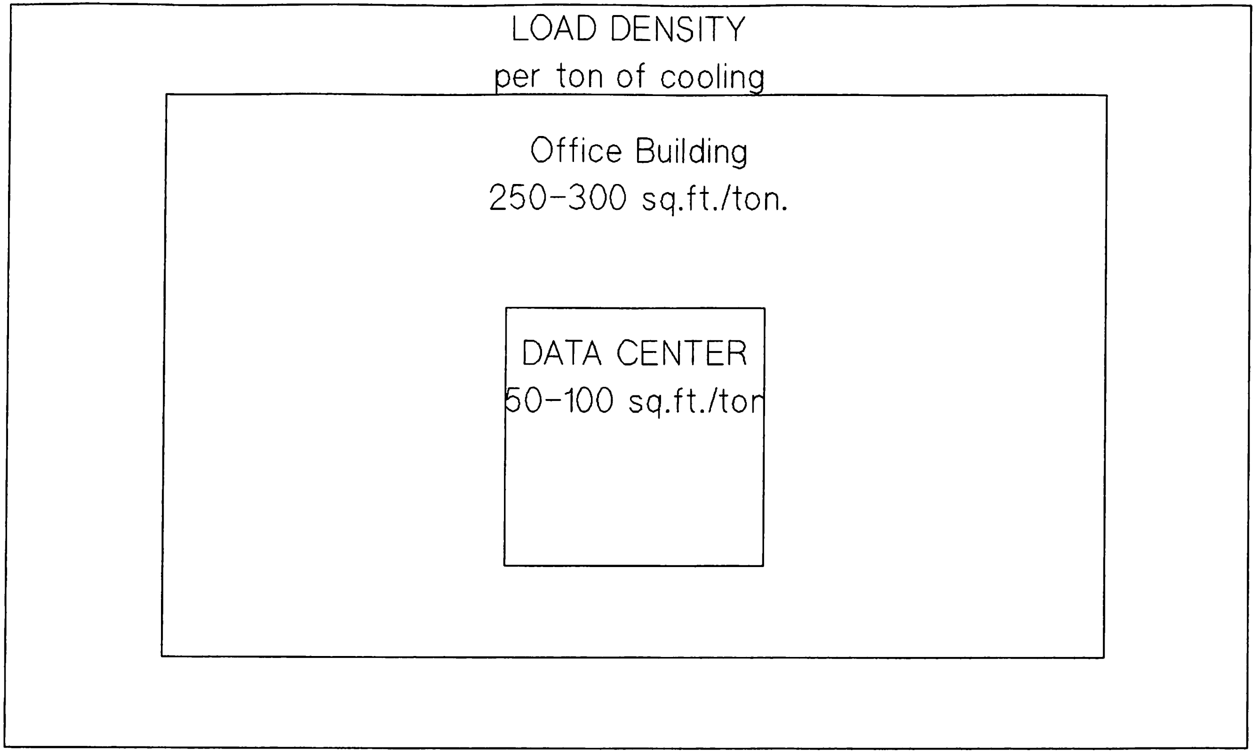
Building a Home...0119

FIGURE 3



Building a Home...0119

FIGURE 4


LOAD DENSITY
per ton of cooling
Office Building
250-300 sq.ft./ton.
DATA CENTER
50-100 sq.ft./ton

Building a Home...0119

FIGURE 5

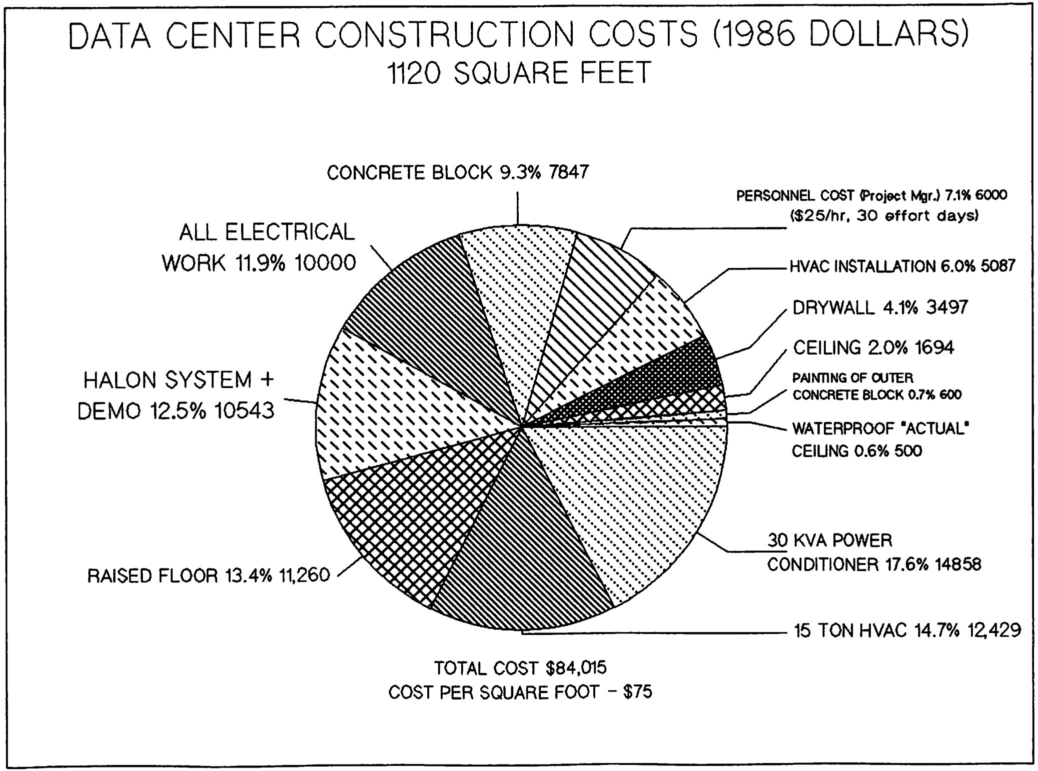

DATA CENTER CONSTRUCTION COSTS (1986 DOLLARS)
1120 SQUARE FEET
CONCRETE BLOCK 9.3% 7847
PERSONNEL COST (Project Mgr.) 7.1% 6000
($25/hr, 30 effort days)
ALL ELECTRICAL
WORK 11.9% 10000
HVAC INSTALLATION 6.0% 5087
DRYWALL 4.1% 3497
CEILING 2.0% 1694
PAINTING OF OUTER
CONCRETE BLOCK 0.7% 600
HALON SYSTEM +
DEMO 12.5% 10543
WATERPROOF "ACTUAL"
CEILING 0.6% 500
30 KVA POWER
CONDITIONER 17.6% 14858
RAISED FLOOR 13.4% 11,260
15 TON HVAC 14.7% 12,429
TOTAL COST $84,015
COST PER SQUARE FOOT - $75

FIGURE 6

Reflection versus AdvanceLink

R. C. Steambarge, Jr.
Lockheed - EMSCO
2400 Nasa Road 1
Houston, Texas 77058

and

John E. McLean, Jr.
JMA Technology, Inc.
P.O. Box 570727
Houston, Texas 77257

I. INTRODUCTION

Today, personal computers are being attached to mini-computers in greater and greater numbers. These require that some type of conductivity be available between the two systems. This is usually done via a terminal emulation package. This paper is concerned with comparing two of the oldest product lines of terminal emulation packages for the Hewlett Packard 3000 series of mini computers. They are Walker Richer & Quinn, Inc. (WRQ) R7+ and Hewlett Packard's (HP) AdvanceLink.

The comparison was between HP AdvanceLink version B.01.00 and on WRQ's R7+ version 3.0. These were chosen since they are the latest product available, have the most recent enhancements and are similar in capabilities.

The major topics to be covered in this paper are

- Requirements of running the programs
- Display capabilities
- Command language
- File transfer
- Configuration control
- Various miscellaneous topics

These topics were selected since this is where most of the confusion can occur in an evaluation.

II. CONFIGURATION REQUIREMENTS

Each software package was compared can run on a IBM PC or compatible. This also includes the XT and AT class machines. The minimum requirements are as follows:

Requirement	AdvanceLink	R7+
Operating System	DOS 3.1 or Up	DOS 2.0 or Up
Memory Usage		
Alpha Mode	256 KB	256 KB
Graphics Mode	512 Kb	310 KB
Disk Usage		
Floppies	2	1
Floppies & Hard Disk	1 & 1	0 & 1
Display Adapter	Monochrome	Monochrome

It should be noted that the memory requirements contain the default memory size for the display memory. This number can change if the amount of display memory is adjusted by the user.

III. DISPLAY CAPABILITIES

This section will into look the display capabilities that these two products support. The AdvanceLink and R7+ do graphics emulation.

Display Capabilities	AdvanceLink	R7+
Graphics Resolution		
Horizontal	640	640
Vertical	350	480
Highest type of monitor	EGA	VGA
Supported by MS-WINDOWS	NO	YES
Full Tektronix's GIN support	NO	YES
Support HP2648A commands	NO	YES
132 column mode	NO	YES
No. of characters scrollable horizontally	162	9,999
User definable color selection	YES	YES
Combine underline and inverse video	NO	NO
Number of vertical lines	25 or 26	25 or 26
Amount of display memory	Max 64 KB	Max 700 KB

There are several notes that must be taken into account when reviewing the above table.

The highest graphics resolution has much to do with the capabilities of the hardware that is supported. Since AdvanceLink at this time does not support the Super EGA and VGA adapters the resolution is limited to that range.

The combination of the underline and inverse video is available when using monochrome adapters. On the color adapters this is a hardware design limitation not a software limitation.

The two products will on run on Monochrome adapter or Color Graphics Adapter,but only allow 25 lines to be displayed. Both products will allow you to select if two lines or one line is to be displayed for the function keys. But AdvanceLink will not remove the 24 line if an application wants to display a line of data there. Where as, R7+ will scroll the labels down to the 25th line and than display the data.

The 700 KB of display data that R7+ supports requires that an expanded memory system (EMS) driver be installed.

The 132 column mode supported in R7+ can be programaticly selected by using the ZENTEC (DIRECT 825) escape sequence. This is not the new escape sequence that is used by HP 700 series of terminals.

The difference in the number of characters that are horizontally scrollable is purely a design decision. AdvanceLink's limit of 162 characters is the same as that which the HP 2626A and 2626W had, thus sticking closer to a pure emulation. Whereas R7+ gives a greater range, thus giving an added value of displaying those extra wide printed reports.

R7+ support of MS-Windows is limited to being able to be selected from the menus and suspending all other programs in background. R7+ can be used in either alpha and graphics mode. AdvanceLink can also be run under MS-Windows but only in alpha mode. The graphics mode is to large to run under MS-Window manager. This is why the response to that item was "NO" since it can not run under both modes.

The HP2648A command support is to handle some special graphics commands which are unique to that terminal and HP2647A. This is true for the read cursor position and wait. This allowed the user to move the cross hair cursor around the screen, and then terminate it with a key and then send the coordinates back to the host computer. This is especially handy for interactive graphics.

There are two features that are not available in either one of the products. The first feature is a raster image transfer of the graphics memory. This transfer can be back to the host or to a local mass storage device. This can be very help when a complex drawing is need to just edit a couple items since the whole drawing would not have be retransmitted. The second was the autoplot feature for line drawings. This not a big item since more powerful graphics programs are available for the personal computer.

IV. MACRO/COMMAND LANGUAGE

HP AdvanceLink and WRQ R7+ both provide a command language allowing the execution of repetitive command sequences from a file. The commands allow the user to set parameters for terminal emulation, file transfer, error handling, display control and user interaction. This section will take a look at this feature.

AdvanceLink command language has 96 commands that can be used within a command file. R7+ has 83 commands but the "SET" parameters can expand this to 121 because there are 38 parameters for the command. This is an indication that both programs have a rich and varied command language.

The two programs took different approaches in implementing this feature. This is shown by the syntax rules that apply to both programs.

1. The Language Syntax

The AdvanceLink use a macro type language. The following summary table will summarize the syntax rules.

- All commands can be abbreviated to the smallest number of characters which make it unique.
- Each command and variable must begin with a "&" (ampersand).
- Commands, functions, variables, DOS and MPE filenames can be in any case.
- All parameters are started by "&P" and followed by digits 0 thur 30.
- Text is limited to 80 characters even if it is a quoted string or variable (&P).
- All functions must start with a "&" (ampersand).

- Comments in a command file must start with "&!" in column 1 and 2. This causes the entire record to be ignored by the command processor.

- Control and escape characters must be started with a circumflex or "hat" (^) and followed by the ascii character which composes it.

- Strings within quotation marks are optional. If the string is to include a space, comma, ampersand it must be enclosed in quotation marks. They may be either double or single quotation marks.

- Commas must be used to separate variables and parameters.

WQR R7+ language is based on an interruptive language similar to the BASIC language. The following table summarize the syntax for the command language.

- All commands can be abbreviated to the smallest number of characters which make it unique.

- Commands, functions, variables, DOS and MPE filenames can be in any case.

- A command must be on a single line of the command file.

- All parameters are started "V" followed by digits 0 thur 10. Can be extended to 800 characters.

- Text is limited to 80 characters.

- Comments in a command file are denoted by preceding them ";". The comment can start in any column and will end at the termination of the current line.

- Control and escape characters must be started with a circumflex or "hat" (^) and followed by the ascii character which composes it.

- Strings must be placed in quotation marks. They may be either double or single quotation marks. This only applies to strings that are being used as a variable.

- If the "^" is not to be interrupted as the start of an escape sequence, it must be entered in as "\^". If the back slash is also to be entered, it must be entered this way "\\".

- Each command word or function must be separated by at least one space.

2. Activating or Executing Commands

Both programs have several methods of starting the execution of a command or command file.

R7+ allows the user to enter a command one of three ways:

1. The first is to press the F5 key from the main Reflection menu. This will display the input command line.
2. The second method allows the command line to be displayed while anything is displayed on the function keys. To access the command line, just press ALT-Y.
3. The third method allows a command file to be executed from the run line when R7+ is started. At this same time you may also pass to the command file starting values for some of the variable 0 thur 9.

The first two access methods allow for executing a command file using one of the commands available to start executing the command files. Alternately, the name of the command file can be typed in on the command line followed by a carriage return.

AdvanceLink allows the command language to be accessed in several ways.

1. Press the ADVLNK command key "F3" on the main AdvanceLink command function keys. A selection then has to be made if an AdvanceLink command file, a DOS command or a AdvanceLink command is be processed. The selection is made by pressing one of the appropriately labeled function key.
2. The second method allows a command file to be executed from the run line when AdvanceLink is started from a batch file.
3. Typing the file name at the main menu and not in terminal mode.
4. Processing the special autoexec files. They are named LOGON.SLK and COMMAND.SLK. These are processed in order their respective order.

3. Comparison of the Command Languages

This section of the paper will take a one to one comparison of some of the more commonly used commands in AdvanceLink and R7+. The commands will be reviewed by in functional groups. Those commands that do not have any corresponding function in one for the programs will be mark by "(n.a.)" in that column. If a command sequence will do the same function that sequence will be shown.

FILE TRANSFER COMMANDS	AdvanceLink	R7+
Vendor Transfer	&DSCOPY &DSCOPY	SEND RECEIVE
Xmode Transfer	&XSEND &XRECEIVE	XSEND XRECEIVE
Kermit Transfer	(n.a.) (n.a.)	KSEND KRECEIVE
Raw ASCII Transfer	&SENDFILE &HP3000 &LOGON	TRANSMIT (n.a.) OPEN + LOG
PC File Backup	&BACKUP &RESTORE	BACKUP RESTORE

TRANSFER CONTROL COMMANDS

	AdvanceLink	R7+
Transmission Block Size	&BLOCKSIZE	SET BLOCK-SIZE
Block Retry	&RETRY	SET RETRY-LIMIT
Startup Timeout	&TIMEOUT	SET RECEIVE-TIMEOUT
Tab Expansion	&TAB	SET TABS-TO-SPACES
Tab Compression	(n.a.)	SET SPACES-TO-TABS
Strip eighthBit	&7BIT	SET ISO7-TO-ROMAN8
Leave eighth Bit	&8BIT	SET ROMAN8-TO-ISO7

TRANSMITTING DATA

	AdvanceLink	R7+
Send a String with CR	&SEND	PTRANSMIT
Send a String without CR	&QUICKSEND	TRANSMIT
Break Signal	&SENDBREAK	BREAK
Logon to HOST	&ATTENTION	PTRANSMIT ""

CONFIGURATION COMMANDS

	AdvanceLink	R7+
Baud Rate	&BAUD	SET BAUD
Communication Handshake	&HANDSHAKE	SET ENQ-ACK SET RECEIVE-PACING SET TRANSMIT-PACING SET DSR-REQUIRED
Parity	&PARITY	SET PARITY
Select a Port to Use	&PORT	SET DATACOMM-PORT
Change Emulation	&PERSONALITY	SET TERMINAL-CLASS
Maintaining Configuration Files	&MAKECONFIG &USE	SAVE LOAD

COMMAND FILE CONTROL

	AdvanceLink	R7+

Conditional Branching	&IF
	&ELSE
	&ENDIF
		IF
		ELSE
		ENDIF

Unconditional Branching	&GOTO &LABEL	GOTO : (name)

Subroutines	&CHAIN	CHAIN
	&EXECUTE	INVOKE
	&RETURN	RETURN
Cleanup Subroutines	&FINISHFILE	ON EXIT
Command file termination	&EXIT	EXIT
	(n.a.)	STOP

USER INPUT AND OUTPUT

	AdvanceLink	R7+
Writing to CRT	&MSG	DISPLAY
	&VERIFY	VERIFY
	&TYPE (DOS)	TYPE (DOS)
User Input	&INPUT	ACCEPT
	&SINPUT	(n.a.)
	&QINPUT	(n.a.)
Host Input	&READDC	READHOST
Disk Commands	&DTEST	(n.a.)
	&DPURGE	(n.a.)
	&DRENAME	RENAME
	&DCOPY	COPY
	&DOPEN	OPEN
	&DREAD	READ
	&DWRITE	WRITE
	&DCLOSE	CLOSE
	&EOFTST	EOF (IF)
Timing Commands	&WAITDC	WAIT ... FOR
	&PAUSE	HOLD
	&WAITCLOCK	WIAT UNTIL
	&WTIMEOUT	WAIT
	&EXPECT	WAIT ... FOR ""

BUILT-IN FUNCTIONS

	AdvanceLink	R7+
Math functions	&ASSIGN	= or LET
	&ADD	+
	&SUB	-
	&MUL	*
	&DIV	/

String function	&UPPER	&UPPER
	&TRIM	&PACK
	&LENGTH	&LENGTH
	&POSITION	&FIND
	&ASCII	(n.a.)
	(n.a.)	&LOWER
	(n.a.)	&MID
	(n.a.)	&VALUE
	(n.a.)	&TIME
	(n.a.)	&DATE
	(n.a.)	&DIR

With the command structure as large as the these two programs support it is helpful that an on-line help facility be available. AdvanceLink does not provide this type of facility. R7+ does provide this feature. R7+ help is available from the command line by just typing HELP followed by the command name. This can be helpful when you need to know the parameters of a command in detail.

The two products have a feature that makes available to the user or programmer all of the commands from the host computer. the both did it differently. AdvanceLink needs to be set into a mode that will allow it to receive the commands from the remote computer. This is done via "&HOSTCONTROL" command. This can be quite announce if you do not have control over the personal computer. R7+ did it with a special escape sequence which is constantly processed. This makes quite easy to run commands on the remote PC once the communication line is established.

The command structure of both products is very similar in the number of commands and type. There are some areas that are different. To the overall use of the commands this does not amount to much in the final analysis. There are others that do make big difference.

4. Error Recovery and Testing

AdvanceLink and R7+ both feature a mechanism for error recovery and analysis. The big difference is that R7+ provides more details and allows it to be used in more places.

The first error recovery method is using a command called CONTINUE. This is similar to the command found in the HP3000 stream facility. If a major error is discovered that would abort the command file, this flag causes the command file to keep going. With this flag set it is up to the command file designer to handle all error conditions and testing.

The second method of error recovery is using special event flags which each program provides. These are used with the &IF ... &ELSE or the IF ... ELSE commands, in the two products.

AdvanceLink provides the user with four boolean (TRUE/FALSE) condition codes. These are:

&TIMEDOUT	That a command was not completed within a specified time period. This is used in conjunction with &WTIMEOUT.
&DOK	Indicates that the must recent disk access was done properly or not. This is used with &DTEST.
&TOK	Indicates the success or failure of a file transfer. This is used with &DSCOPY, &XSEND, &XRECEIVE.
&AOK	Indicates that everything that is all right with the commands that have been executed. This is used with &ATTENTION, &CALL, &MCONNECT, and &VCONNECT.

The above condition codes can only be used with the commands that are mentioned with them. There are no codes to give any additional information about the error that did occur. AdvanceLink does not provide any other code or reserve words for these functions.

R7+ provides the user with six boolean (TRUE/FALSE) condition codes. They are

DCD	Checks for the carrier detect being present. Used at any time.
EOF	Checks for the end of file. Used with the READ command.
ERROR	Checks for any error condition to have occurred with the previous command.
EXIT	Checks for a named file being present.
FDG	Checks to see if R7+ is in the foreground operation.
FOUND	Used to determine if a string or condition has occurred. This command is used with HOLD, WAIT and READHOST.

The general error condition code "ERROR" can be expanded to give more detail on the type of error by using the integer operator ERROR-CODE. This will return a number that is for a particular error that has happened. Some of the error return codes can only be used with certain commands. The areas covered are for file transfer, reading and writing to disk files, serial interface, the disk backup facility and the command syntax.

5. Variables

In the section on the command syntax, there was some mention of variables that are supported by each command group. This capability allows the command file to become quite flexible at handling various communication tasks. The variables can be used to develop an interface that can be quickly understood by an user who is not a communication expert.

The variables can be used in most of the commands. The assignment of variables is handled differently between the two programs.

AdvanceLink requires that everything be done with the &ASSIGN command. This is even true for the concatenation of strings. The assignment of numeric values requires the data to be placed between a set of quotes. There is no implicit numeric conversion available.

R7+ requires that the LET command be used to assign data to the variables. Within a command a quoted string and string can be concatenated with the use of the "&" between them. Numeric values do not need to be placed in quoted strings before being loaded into a variable. The variables can be used inside of a quoted string by using the "$" version of the variable name.

You should remember that only 80 characters can be stored in each variable for both products.

V. COMMUNICATION SUPPORT

1. File Transfer

This is the one of the most widely used features of the terminal emulation software, outside of the actual emulation itself. A file transfer capability is provided by both products. The number of transfer protocols that are supported by the two programs differs.

Transfer Protocols	AdvanceLink	R7+
Vendor Protocol	YES	YES
Kermit	NO	YES
XMODEM	YES	YES
TEXT Files	YES	YES
Hard Disk Backup/Restore	YES	YES

The number and type of protocols that are supported by the packages only become important when different host computer are used with the same PC.

AdvanceLink's vendor provide protocol can only be used with HP computer system. WRQ R7+ vendor provide protocol can be used with HP, Digital, UNIX. These protocols have been found to be faster than the public domain protocols because they can be tailored for use on a particular operating system or hardware platform.

Kermit and Xmodem are two of the most widely used file transfer protocols in the PC market. It is of great benefit to have both protocols in one package. This assures that the user can take to almost any computer system.

Xmodem is usually required by most bulletin boards. This is also required if you are using any of the major information services.

KERMIT is very helpful in those installations which are in a multiple computer vendor shop and the user must have control over the file transfer program on the host computer. Kermit's protocol and command structure allows you the ability to configure exactly how the file transfer is done. It is also useful in the places where the data format is different between computer systems. There is a capability to transfer binary data if need be. Another feature of the Kermit protocol is that it allows for wild card file transfer.

The last file transfer method is text files. These are usually required to boot up the vendor host protocol program. It is also an easy way of dumping text data into some type of editor without a whole lot of trouble.

Both AdvanceLink and R7+ have a set of commands to backup and restore files from the personal computers hard disk. This is can be also looked at as a file transfer method for the PC. Both programs use the vendor provided protocol to facility the transfer of the files between the host computer and the PC.

One of the nice things about this feature is that it can backup a whole hard disk unattended or just a selected directory and its subdirectory if required. Both programs provide a log of what files where backed up. The AdvanceLink log is just a little encrypted whereas the R7+ log can be read as is.

WRQ provides the user with two utilities to work with backup file on the host computer. The first program HPDIR will generate a directory listing or log of what files are in the backup file. This is nice to have since you do not need to bring the file back down to get this information. The second program, XTRACT, will remove a single file out of the backup file and place into a file on the host computer.

The AdvanceLink does have a feature that will generate default file name for the host computer if you select it to do so. This could be quit handy for unattended backups.

One problem with both programs when using this feature is that the user must determine the number of records that the backup file on the host computer is to have. If the file is too large, you will have to reduce the file size by some means such as using MISER out of the contributed library.

2. Communication Hardware Support

The next thing to look at is the communication hardware which the two products support.

Communication parameters	AdvanceLink	R7+
Communication speeds		
Lowest Baud rate	110	110
HIghest Baud rate	19,200	38,400
Number of Serial Ports	2	8

For most HP 3000's the 19,200 baud rate is the best it can do. The higher serial communication speeds which are being produced today, the faster speed will be greatly appreciated. This is true for those applications which paint the entire screen.

There are always those times when having the laser jet attached and running two sessions at the same time would be nice. That takes either 3 ports or a switch box. This type of setup is becoming more and more common. This is especially true those individuals running Microsoft WINDOWS or going to run Microsoft OS/2. This why IBM in the PS/2 allows the support of eight serial communication ports.

VI. CONFIGURING THE TERMINAL

Since each user has different way of working, it is important that terminal emulation software be as changeable as possible. That is what this section of the paper will look at.

1. Ways of Setting the Configuration

AdvanceLink and R7+ both allow the user to configure the software in several ways. They are as follows

- Screen Menus access from the function keys
- Using certain commands in the command language
- Loading the configuration file at run time

The ability to set the configuration from the command language is helpful when certain things are required to be changed for particular task that is a little different then the basic configuration file. This is a necessity for those PC's that are used by a group of users and critical parameters must be set to particular values for the application to work.

The ability to save the configuration in a file and recall it allows a user to have several different configurations that can be used for different tasks. One task that would benefit from this is to receive and send telexs from a telex service and then turn around and send it up to the host computer for printing.

2. Changing the Keyboard Layout

In recent years, applications have moved from the mini-computers to the PC and vice versus. Also the keyboards of the PC's do not match those of the traditional terminals. Users of these applications or terminals would like to remap their keyboards to feel the same as the way they are use to working. Some terminal manufactures are allowing a keyboard remapping feature. So a terminal emulator should also be able to do this.

This version of AdvanceLink does not provide this feature.

R7+ version 3.0 does have this capability to remap the keyboard. This is done by the user building a script file external to R7+ with their favorite text editor or word processor. This is than compiled into the configuration file using an utility program, called KEYMAP.EXE, that is provided by WRQ. There are some keyboard remap files on the distribution diskettes.

3. Supporting an External Printer

Both products support external printers. This support takes on the form of expanded, compress and graphics screen dumps. The two products allow for other printers to be supported in this mode. This allows for the printers to be parallel or serial.

AdvanceLink does allow other printers to have their escapes sequences to be defined. This is not allowed with R7+, but can be accomplished using certain commands to setup the print.

R7+ does allow the use of a driver from the program. This would allow the escape sequence to be converted or handled if need be.

4. Expanded Memory

With the introduction of the Lotus Intel Microsoft (LIM) expanded memory system (EMS), more individuals are extending the memory range beyond the 640 KB limit of DOS. This capability would free up memory in the standard DOS memory thus allowing other programs to run. This feature is not available in AdvanceLink but is in R7+. Using this feature with R7+ would reduce the main memory requires by 64 KB.

VII. Miscellaneous Topics

1. Other terminal emulation availability

With a terminal emulation software the question always comes up "Does it provide other terminal emulation?" This is true for both programs and the other products by the two companies to provide additional emulation.

Terminal Emulation	AdvanceLink	R7+
VT 52	YES	YES
VT 100 or 102	YES	YES
VT 200	NO	YES

The VT 200 emulation that is available with R7+ is a subset of the actual terminal. If a full VT 200 emulation is required, then you can use WRQ R2 or R2+ product which provides the same command set as R7+. The major items left out of the product are not a complete set of user defined keys (UDKs)and multinational character sets are not supported.

If you are in a shop that has VAX's and need the support of VT 341 terminals, then WRQ can supply you with R4 or R4+. This could make your support task simpler since the command language is the same for all products.

2. Availability to Run on other Hardware Platforms

This was looked at for two reason how protected is the expense of user training where a hardware platform is changed. And can the support of one product be spread over several platforms.

This becomes a major concern for companies who are using more than one type of personal computers. Many organizations are getting IBM PC and compatibles for business application and Apple Macintosh for desktop publishing and word processing.

Platform Availability	AdvanceLink	R7+
IBM PC	YES	YES
IBM PC Compatibles	Maybe	YES
Apple Macintosh	NO	YES
HP 150	NO	YES
HP Vectra	YES	YES
HP 110	YES	YES

It should be noted that IBM compatibles for the AdvanceLink is a maybe. That means it is trial and error to determine if the product works on that particular PC.

3. Multitasking of the Program

A feature that was found to quite useful was R7+ ability to continue processing in the background and allow another process to be interactively used in the foreground. AdvanceLink does not have this capability. This can be used while a file is being transferred down to the PC at the same time you are editing on another file.

This should not be confused with the capabilities to execute another program from within the two programs. This allows only the child process to be active until it is terminated and control is returned to the parent process.

4. Forms Cache

During the investigation phase of the this paper, an interesting fact was found. The fact is that AdvanceLink does not support forms cache and that R7+ does. The forms cache feature is one of HP big sells point on the their HP 2624B and 2394 terminals. This feature may have been left out because of marketing reasons. R7+ has taken this feature and expanded it to allow 255 forms to be downloaded to the PC.

5. Some Vendor Supplied Utilities

AdvanceLink does come with an IBM PC version of PAM. It allows any encrypt commands files to have passwords on them which PAM will then prompt for. It provides a somewhat friendly user interface which the documentation says is very slow. It does allow step by step prompts for setting of the application to be run. It does come with a visual file manager.

Both products come with a translator program for the other command files as long as they are in ASCII form.

An encryption program is supplied for the command files. The encrypted command file can be protected by a password which is optional.

R7+ does provide a set of programs for transferring data between PC's directly. There are versions for the HP150, IBM PC, Wang and Digital Equipment Corporation personal computers. These rates can be at 38400 if the hardware is fast enough and the program being executed can support it.

6. Support and Manuals

What would a product evaluation be without some mention of the documentation and the support that is provided by the vendor.

AdvanceLink comes in a single binder. Each of the major sections are tabled for easy selection. It does have a comprehensive index and table of contents.

The support of product in through HP and is charged at an hourly rate. So have your credit cards ready.

R7+ comes in three spiral bound books. The books are the user manual, technical reference and command language. This allows you to just use those parts which you need to do the job at hand. Each books index references the other manuals. The table of contents just covers the material for that manual.

Support for the product is provided by the WRQ support line and a group of highly qualified independent dealers. The charge for this is free.

VIII. Summary

As can be seen, that both products provide allot of features. The selection of which one to use can be difficult. You, the user, must look at the job to be done and then match the product.

TITLE: Distributed Applications and the
Communication Link

AUTHOR: Doug Walker

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 0126

A PROGRAMMING ENVIRONMENT IN C

by Larry Simonsen and Keven Miller
VALTEK INCORPORATED
Moutain Springs Parkway
Springville, UT 84663

All programmers have a style of their own. This style is reflected in the data structures they create. The C programming language provides the means to create a programming environment which reflects these dynamic differences of style. This paper describes the programming environment (software) of VALTEK's C programming project.

The C programming language has relatively few data structures and directives. However, each is well defined (though sometimes confusing to those with experience in other languages). We have taken these few data structures, with the capability of creating others, and have built a unique programming environment.

Much like a building is built on a foundation, we (as others can) have built on features defined by the ANSI draft standard.

Our programming block structures can be compared to the levels of the OSI networking structure much like a software network. These levels are as follows:

LEVEL	NAME	SOFTWARE APPLICATION	
7	APPLICATION	MULTI-TRANSACTION ENVIRONMENT	
6	PRESENTATION	TRANSACTION DEFINITION	
5	SESSION	DATA ITEM HANDLER updimqoh	
4	TRANSPORT	USER ENVIRONMENT strinput	USER DBMS INTERFACE imserial
3	NETWORK	DBMS / LANGUAGE SPECIFIC dbget, dbopen / strcopy, float	
2	LINK	OPERATING SYSTEM / FILE SYSTEM fread, dateline	
1	PHYSICAL	HARDWARE OF THE SYSTEM load, store, add, branch	

As in the OSI model, where each layer provides a specific function and can be replaced if the interface to adjacent layers is maintained, so the software model can support this type of procedure/level replacement. The interface between layers needs to be well defined and a given layer should not skip layers to call procedures from lower levels.

Design criteria of new programming environment:

1. System structure (global data) of about 400 bytes so that no global data storage is used in routines. All procedures can then be placed in SL libraries.

2. Each dataset of every database will have a structure corresponding to its record layout. These dataset structures, combined with organized constant definitions, become our data dictionary (i.e. the compiler would know about the dictionary and its structure would appear in program listings).

3. For each command the same database record would not be read twice.

4. Records could be resorted several times without reselecting or rereading the database (local file I/O is faster and does not impead other processes).

5. All database fields can be classified as cost, hours, percent, price, quantity, string, and summary price (if you have others please let us know).

   Each field will have defined field descriptors of ITEM, PROMPT, TITLE, LEN (for arrays), WIDTH, PREC, FORMAT, EPSILON, and MAX. Another possible addition is MIN. Our environment treats the above types as if they were fields and defines the PROMPT, TITLE, WIDTH, PREC, FORMAT, EPSILON, and MAX for each as shown below:

   ```
   #define  costPROMPT  “cost”
   #define  costTITLE   “COST”
   #define  costWIDTH   12
   #define  costPREC    3
   #define  costFORMAT  “f”
   #define  costEPSILON .001
   #define  costMAX     99999999.999
   ```

   These are then used to define the database fields. For example:

   ```
   #define  poucWIDTH   qtyWIDTH

   #define  itnoITEM    1
   #define  itnoPROMPT  “Part Number”
   #define  itnoTITLE   “PART  NUMBER”
   #define  itnoLEN     18
   #define  itnoWIDTH   18
   #define  itnoPREC    18
   #define  itnoFORMAT  “s”
   ```

6. The most commonly passed parameters to procedures will be the first four.

7. No routine should be longer than 150 lines. These small, easy to prove correct routines are used to build larger procedures. Small procedures can be used elsewhere.

8. Normalize your code as you would your data.

9. Adopt and build on debugging ideas presented from others:

   David Greer - Las Vegas. Build a test suite from user's failure data.

   Denis Heidner - Las Vegas. Save a copy of data that causes failure (snapshot).

   Bruce Toback - Detroit. Be able to (without recompiling) turn on debug printout information.

10. Heavy user envolvement as system evolves. Rely heavily on the small core of knowledgable users. Trust their comments as to user interface and data, realizing, of course, that you're ultimately responsible.

Below is a sample of one of our reports which utilizes the levels we've discussed. Our code has the appearance of a fourth generation language. When compiled it generates considerably faster run time code than our old third generation software.

```
#include  "env.h"

#define  cmdNAME  rg100
#define  Dispatcher  gl

#include "accmas.h"  /* def. of   accmas database record */

SUBROUTINE( cmdNAME )
{
  VALBASE_DEF;

/*
REPORTS ARE FORMATED AS

HEAD
TITLE
LINE
LINE
TRAIL
TITLE
LINE
TRAIL
FOOT
<<page break>>

if you do not have a given type of format then delete that format.  each of
these types of formats may be multi-lined.
*/

#define   hdfmtVARMAX        20
#define   hdfmtMAX           1000
```

```
#define   DETAILLINESMAX    2 + 2

#define   ttl0fmtVARMAX      4
#define   ttl0fmtMAX         350
#define   ttl0fmtNUM         2

#define   ln0fmtVARMAX       11
#define   ln0fmtMAX          100
#define   ln0fmtNUM          3

  char   datestr[28];

  UNIT_DEF( unitout, DETAILLINESMAX );
  i2   unitoutbrkflag;

  FMT_DEF ( unitout, UNITFMTNUMHEAD, hdfmtMAX,  hdfmtVARMAX );
  FMT_DEF ( unitout, ttl0fmtNUM, ttl0fmtMAX, ttl0fmtVARMAX );
  FMT_DEF ( unitout, ln0fmtNUM, ln0fmtMAX, ln0fmtVARMAX );

#define  sortvarMAX          2

  char    key0prompt[] = accseqPROMPT;
  char    key1prompt[] = accnumPROMPT;
  i2     sortmask;
#define  fieldsMAX           11

/* define here the db record buffers */
  DEFPTR( accmas_t,  accmasptr);
  db_rec_t accmascurrent, accmashighwater;

/*------------------ Program Variables ----------------*/

  data_t      *data;
  i2          datanum;
  fieldlist_t  cmdfield[1+11] = { (char*) 11, 0 };

/*      TEMP CALCULATION         */
  char   outputacctype[acctypWIDTH+1];
  char   outputdesc[accdesWIDTH+10+1];
  char   outputstmtype[accsttWIDTH+1];
  char   outputnorbal[accnorWIDTH+1];

#pragma   page
/*------------------------------------------*/
/ *                                        * /
/ *             CODE                       * /
/ *                                        * /
/*------------------------------------------*/
  entervoidroutine( stringof(cmdNAME) );

  _dateline(datestr);
  UNIT_INIT(unitout);
  fmt_init (unitout, UNITFMTNUMHEAD, UNITFMTTYPEHEAD);
  fmt_init0(unitout, UNITFMTNUMFOOT, UNITFMTTYPEFOOT);
  fmt_init (unitout, ttl0fmtNUM, UNITFMTTYPETITLE);
  fmt_init (unitout, ln0fmtNUM, UNITFMTTYPELINE);

  data = env->data[ datanum = recalloc(0l, 10) ];
  dataalloc(datanum, fieldsMAX, sortvarMAX);

/*custom*/
  /* add the formats */
  datafieldsort(datanum, accmasptr->accseq,
    SORTASCENDING, SORTREAL, key0prompt);
  fmtvar(unitout, ln0fmtNUM, accmasptr->accseq,
    fmtfield(accseq));
  fmt(unitout, ttl0fmtNUM, accseqTITLE, accseqWIDTH);

  datafieldstrsort(datanum, accmasptr->accnum, accnumLEN,
    SORTASCENDING, SORTBYTE, key1prompt);
  fmtstr(unitout, ln0fmtNUM, accmasptr->accnum,
    fmtfield(accnum));
  fmt(unitout, ttl0fmtNUM, accnumTITLE, accnumWIDTH);

  datafieldstr(datanum, accmasptr->accdes, accdesLEN);
  fmtstr(unitout, ln0fmtNUM, outputdesc,
   fmtfield(accdesindent));
  fmt(unitout, ttl0fmtNUM, accdesTITLE, accdesindentWIDTH);

  datafield(datanum, accmasptr->acctyp);
  fmtstr(unitout, ln0fmtNUM, outputacctype, fmtfield(acctyp));
  fmt(unitout, ttl0fmtNUM, acctypTITLE, acctypWIDTH);

  datafieldstr(datanum, accmasptr->accstt, accsttLEN);
  fmtstr(unitout, ln0fmtNUM, outputstmtype, fmtfield(accstt));
  fmt(unitout, ttl0fmtNUM, accsttTITLE, accsttWIDTH);

  datafield(datanum, accmasptr->acccla);
  fmtvar(unitout, ln0fmtNUM, accmasptr->acccla, fmtfield(acccla));
  fmt(unitout, ttl0fmtNUM, accclaTITLE, accclaWIDTH);

  datafield(datanum, accmasptr->accnor);
  fmtstr(unitout, ln0fmtNUM, outputnorbal, fmtfield( accnor));
  fmt(unitout, ttl0fmtNUM, accnorTITLE, accnorWIDTH);

  datafield(datanum, accmasptr->acctol);
  fmtvar(unitout, ln0fmtNUM, accmasptr->acctol, fmtfield(acctol));
  fmt(unitout, ttl0fmtNUM, acctolTITLE, acctolWIDTH);

  datafield(datanum, accmasptr->accskp);
  fmtvar(unitout, ln0fmtNUM, accmasptr->accskp,
    fmtfield(accskp));
  fmt(unitout, ttl0fmtNUM, accskpTITLE, accskpWIDTH);
  datafield(datanum, accmasptr->accind);
  fmtvar(unitout, ln0fmtNUM, accmasptr->accind,
    fmtfield(accind));
  fmt(unitout, ttl0fmtNUM, accindTITLE, accindWIDTH);

  strcpy(data->sort.defaultinput, “1,2”);
  fmtnewline(unitout, ln0fmtNUM);
  fmtnewline(unitout, ttl0fmtNUM);

  unithd(unitout, dateTITLE, 0);
  unithdstr(unitout, datestr, fmtfield(date));
  unithdend(unitout, pageTITLE, 0);
  unithdendvar(unitout, unitout->page, fmtfield( page));
  unithdnewline(unitout);

  unithdcenter(unitout, “G/L ACCOUNT MASTER”, 0);
  unithdnewline(unitout);

  unithdcenter(unitout, “==================”, 0);
  unithdnewline(unitout);

  unithdcenter(unitout, “SORTED BY “, 0);
  unithdcenterstrprec(unitout, data->sort.desc,
    data->sort.desclen, -fmtfield( sortdesc));
  unithdnewline(unitout);

  unitstatus (unitout, DETAILLINESMAX);
#pragma   page
/*-----------------------------------*/
/ *                                 * /
/ *            SELECTION            * /
/ *                                 * /
/*-----------------------------------*/
/*  open other databases   */

  while (unitselect( unitout ) == INOK) {

    if (!unitopen(unitout, stringof(cmdNAME))) continue;
    headvalpac(unitout, stringof(cmdNAME));

#pragma   page
/*-----------------------------------*/
/ *                                 * /
/ *       GATHER AND SORT           * /
/ *                                 * /
/*-----------------------------------*/

    do {
      sortreset (datanum);
      /* add sort keys if always at begining */
      if ((sortmask = sortinput(datanum)) != INOK) break;
      /* add additional sort keys if always at end   */
      if (!data->reccnt) {  /* no records selected yet */
        env->anyoutstdout =  0;
      if (env->small)
        start_working( &data->work, WORKING_SELECT,
        "accmas", data->current, &env->smallsize);
      else {
        accmashighwater = accmascap(100);
        start_working(&data->work, WORKING_GATHER,
        "accmas", &accmascurrent, &accmashignwatter);
      }

      while(accmascurrent = accmasserial( accmasptr, 10) {
        if (!env->small) im_working(&data->work);

        recwrite(datanum, 200);
        if (env->small) {
          if (data->reccnt >= env->smallsize) break;
          im_working (&data->work);
        }
      }
    }

  if (data->reccnt == 0) {
    unitnooutput( unitout);
    if (IMAJOB) continue;
    break;
  }

  sort(datanum);

#pragma   page
/*------------------------------------------*/
/*                                         */
/*             OUTPUT                      */
/*                                         */
/*------------------------------------------*/
  unitoutbrkflag = 0;
  unitout->page=-1;

  env->anyoutstdout = ( unitout->stream == stdout);

  start_working(&data->work, WORKING_OUTPUT,
    stringof( cmdNAME), &data->current, &data->reccnt);

  while (data->current < data->reccnt && unitoutbrkflag >= 0) {
    recread(datanum, 300);
    im_working(&data->work);

    memset(outputdesc, ' ' ,40);
    strncpy(&outputdesc[accmasptr->accind]
      ,accmasptr->accdes ,accdesWIDTH );
    glacctypedesc( outputacctype, accmasptr->acctyp );
```

```
      glstmtypedesc( outputstmtype, accmasptr->accstt );
      glnorbaldesc( outputnorbal, accmasptr->accnor );

      unitoutbrkflag = fmtwritettl( unitout, ln0fmtNUM,
        ttl0fmtNUM);
      if (unitoutbrkflag >= 0)
        unitskip( unitout,
        (accmasptr->accskp != 10 ? accmasptr->accskp : 9999));
   } /* till all records written */

   unitend(unitout);

     }  while (TRUE);  /* till all sorts done for this output file */
        unitclose( unitout);

  }  /* till no more output */
  recrelease(datanum);

  exitvoidroutine;
}

/*-------------------------------------------------------------------*/

#include "driver.h"
/*-------------------------------------------------------------------*/
```

IDAT: For Dump Analysis and More...

Neil Ferguson
Boeing Computer Services
P.O. Box 7730
M/S K79-30
Wichita, KS 67277-7730

Introduction

IDAT is the name given to the Interactive Dump Analysis Tool. This utility is currently used, mainly by Response Center engineers, to analyze memory dumps. It functions, as its name implies, in an interactive manner, which differs from other methods of analyzing dumps where a paper listing is the only output available. This paper is intended to give users an overview of how the program works, and how you, as an end-user, can use this utility to do some *rudimentary* dump analysis on your own. If you are already skilled enough to do competent dump analysis, then this paper may not present new information for you or develop those skills any further. If, however, you are a skilled programmer, a system supervisor or a system manager, and have always wondered what an engineer looks for when reading a dump, then this paper will probably provide you with some useful information.

Your system has just failed with a SF310. Now what? Do you have an adequate plan to deal with the recovery? Will you have to wait for minutes or even hours for the system to be brought back up? Do you need to take a memory dump? If so, then when you get one, what do you do with the tape? Do you need to phone the Response Center? If you do, what can you be doing proactively while you wait for their return call? Do you know how to perform a rudimentary dump analysis?

These are some of the questions which undoubtedly have been considered by many a System Manager, Operations Manager or even the late night operator. This paper will attempt to enlighten you about one tool which can be used to speed up the process of analyzing a memory dump. This paper *will not* attempt to train you to read *any* memory dump and determine *exactly* what went wrong. That skill (or more aptly, that art) should be left to the personnel at the Response Center. The Response Centers are staffed with specialists who do dump analysis every day. Their knowledge of MPE and their ability to analyze dumps will probably be more honed to perfection than yours.

However, there are a few things which you can do which will speed the process along. After we cover those things, we can consider using IDAT to seek out additional information about your system, while it is still up an running. But first, let's return to the above scenario: the system has failed with a SF310.

Having a total of six years working with the HP3000, my first recommendation is that you always, *always* take a memory dump. Even if you are the only person on the machine and you are testing a new privileged-mode program, I would still take a dump. A Series 70 with 12 Meg of memory will only take about ten minutes to dump, depending on how much virtual memory may be copied to tape. Failure to take a dump could delay you in your efforts to prevent a recurrence of the problem. There is no way to predict whether this failure is a lone freak occurrence or the first in a string of failures which may happen over the course of many weeks or even months. For multiple failures, sometimes the only way to determine the cause is to perform comparisons between the dumps to look for common characteristics.

First, let us make sure that we get a *valid* memory dump. In order to start the dump procedure, at the system prompt, you would enter: DUMP RETURN. Follow the directions that will appear on the system console. The Software Dump Facility (SDF) will then take over the operation of the machine and will create the dump tape. If this process fails, then you may have a more serious problem and you should phone the Response Center for further assistance. The Response Center *may* be able to help you get a valid dump, but don't expect much, especially if the SDF banner has been displayed on your console terminal. This means that the contents of low memory have been copied to disc and that the SDF software has been loaded in its place. If you try to perform another dump, the *original* contents of low memory will be lost during the loading of the *second* copy of the SDF.

Assuming the SDF performs successfully, the dump tape will contain a copy of memory as it existed when you entered DUMP at the system prompt. If you are running an MPE revision of U-Mit or beyond, then you should also have a copy of all data segments which were swapped out to disc at the time of the failure (these segments make up virtual memory) and copies of several system files. Console messages will inform you about the contents of the tape.

After the dump tape is created, the SDF will issue a WARMSTART command. This does not mean that a WARMSTART is being initiated! This command is internal to the SDF software, and functions as though you switched on the machine and entered: START RETURN. After a short pause, you will see the familiar banner requesting which option you wish to use to start the machine. At this point you will have the option to choose a WARM or COOL start. Which option you choose will depend on your particular situation.

After taking the memory dump and restarting MPE, each site will have its own method for system recovery, depending on a wide variety of configuration options and application dependencies. This discussion will not attempt to elaborate on these variations, but instead will focus on what you can do after normal startup procedures have been completed.

You should now have the memory dump tape at hand. If your site can recover without dismounting the memory dump tape, you are that much further ahead of other sites, and you can now make the tape online again, in preparation for use by IDAT. Otherwise, mount the memory dump tape again and make it ready. If your site is such that a night operator has taken the dump, s/he may or may not have the ability and/or permission and/or user capability to run IDAT. Just make sure that the tape is safe until either you or the Response Center has had a chance to analyze it. I would recommend that the tape be labeled with the date and time, MPE revision, and the failure type.

Now we should be ready to run IDAT. The important point is to find the most recent version of IDAT that you can. Most systems will have a version of IDAT in PUB.SYS. It may be named IDAT or IDAT5. Many systems will also have another version somewhere in the TELESUP account. The distribution of the utilities in the TELESUP account is not as uniform as the distribution of MPE, so you may have to look around some. In general, you should find the version whose program file is the largest, since it will probably have the most formatting features.

If necessary, or if you ask someone from HP, you may be able to get a recent version of IDAT so that it will be there when you need it. Having been a former HP SE, and having spent time at the Response Center myself, I used to occasionally download a fresh copy of IDAT on the customer's machine if I could not find a very recent version. However, at 1200 baud, the present size of the IDAT program file is such that this can take as long as forty-five minutes to download, so don't count on HP being able to download a fresh copy to you. Having one already available will speed the dump analysis process along.

NOTE

> As an aside, if you happen to own a Series 5x, you should type Ctrl-B after the SDF has issued its WARMSTART and the system is waiting for you to choose the WARM or COOL start option. Then, enter HALT RETURN at the system prompt. If you allow a Series 52 or 58 to be started from within the SDF subsystem, you will be subject to a bug in the SDF which prevents the Series 5x from enabling the CPU board option which distinguishes a Series 5x from a Series 4x. The result of starting a Series 58 with the SDF WARMSTART command will be a machine that accesses memory *slower than a Series 44.* After entering HALT, you can restart the machine by entering: START RETURN, whereupon you will again see the request for which type of startup you wish to perform. This process ensures that you will have a Series 5x which functions at its full speed.

At this point, the most helpful thing to do is to now run IDAT and copy the contents of the memory dump from tape to disc. I will attempt to give you a cookbook method for the procedure:

```
(sign on to the appropriate user)

:RUN IDAT[.grp[.acct]]

Interactive MPE-IV/V Dump Analysis Tool - 10/27/87
(C) HEWLETT-PACKARD COMPANY 1985

Type 'H' for Help
```

Notice that IDAT contains a HELP facility. It looks for a separate file named IDATHELP, residing in the same group that IDAT resides. If you do not have a file by that name, you may wish to contact someone at HP in order to obtain the proper file. The HELP facility will provide you with the appropriate syntax for the various commands that IDAT uses. Also, the abbreviations used in the output from various formatting and display commands are expanded upon. However, you should be aware that entering the proper commands will not guarantee that you will understand the output you receive. This paper will cover some, but not all, of the capabilities of IDAT.

IDAT's prompt character is a single dash (-). When following the examples below, be aware that you would enter everything *after* the dash.

We can now begin to discuss copying the dump from tape to disc. You can make the analysis process a little easier if you choose meaningful names for the copies of your memory dump(s). If, as in our case, you have a SF310, you can enter the following command:

```
-T SF310,TAPE RETURN
```

IDAT will now attempt to read the memory dump tape that was created earlier. At the console, you will need to reply to the tape request. An example is listed below:

```
:RECALL
THE FOLLOWING REPLIES ARE PENDING:
?10:34/#S842/185/LDEV# FOR "DUMPTAPE" ON TAPE (NUM)?
:REPLY 185,7
```

IDAT will now read the contents of the memory dump tape, transferring the data to a disc file named SF310. If you have a dump tape created by the SDF on U-Mit or beyond, and you are running a sufficiently recent copy of IDAT, the contents of virtual memory will also be copied to a disc file. It will have the name VSF310. (The letter 'V' will be concatenated to the front of the file name provided in the IDAT -T command)

Once the contents of the tape is copied to disc, you can use IDAT to examine the contents of memory by entering the command:

```
-T SF310 RETURN
```

If IDAT sees a corresponding file with the name VSF310, it will attempt to open that file and process its contents as virtual memory. If successful, you will see a message on your screen similar to:

```
        *** VIRTUAL STORAGE IN EFFECT ***

MPE VERSION: HP32033G.A3.01.   (BASE G.A3.01).

********* HALT  17
****SYSTEM FAILURE #310 ; STATUS %140052; DELTA P %013074
```

At this point, you have already helped the Response Center with dump analysis preparation, so you may wish to stop here and wait for the HP Response Center to return any phone calls that you may have placed. When someone does call, s/he can be informed as to the location of the IDAT program and the memory dump file(s) and the engineer will be able to start analyzing your dump immediately.

Of course, one of the advantages to having the dump now residing on disc is that it can be handled the same as any other disc file. It can be :STOREd to tape and sent to HP for further analysis. It can be SQUISHed by any one of several compression programs so that it does not occupy as much disc space (and it can take *a lot* of disc to hold 12 Meg plus virtual memory). You can place other memory dumps in the same group for comparison purposes. You can even get DPAN4/5 to read the dump, if you choose. For the most part, however, there is little that DPAN can tell you that IDAT won't, and IDAT uses *a lot* less paper. A lot of coordinated effort on HP's part has gone into the present system of being able to perform dump analysis over the phone, many times within minutes after a failure has occurred. (See the text covering a brief history of IDAT at the end of this paper for further information)

What we will now consider is: What do you do if you want to know more about the failure before HP calls? The following pages are laid out in an attempt to categorize some of the IDAT commands and they follow a makeshift course. Depending on the output from one command, you may perform other commands. Again, I would like to reiterate that what follows are examples, and they are not intended to be a substitute for being trained to read a dump.

Also, even HP Response Center engineers will vary the choices that they make while reading a dump. There are a few guidelines that can be followed, but, for the most part, you have to resist the temptation to read dumps in a cookbook method. While being trained in dump reading, I was told to remember one rule: Anything, *absolutely anything* could be wrong when the system fails. This means that it could be a hardware problem, an MPE software problem, an application software problem, a hacker who corrupted memory on purpose or even a freak occurrence only attributable to a cosmic ray, or something else equally as obscure.

For each example command which follows, I have listed an objective or action at the beginning, followed by the command to enter, followed by some example output and finally an explanation of what we found and where we should proceed next. For the most part, these examples were taken from actual dumps. Only the names have been changed to protect the innocent.

STEP #1

Format the system registers:

```
-F REG RETURN
```

```
                  *****   REGISTERS   *****

 3/12/88,  7:20PM    MPE 5 (G.A3.01) (BASE G.A3.01) * SERIES 4x/5x  *
*************************************************************************
*   DATA SEGMENT   *  CODE SEGMENT  *MISCELLANEOUS* STATUS = 103027  *
*************************************************************************
* DB BANK = 000013 * PB     = 062130 * X  = 001271 * MODE      = PRIV *
* DB      = 127630 * P      = 071277 * CIR= 030377 * INTERRUPTS= OFF  *
* S  BANK = 000013 * PL     = 106433 * NIR= 000377 * TRAPS     = OFF  *
* DL      = 127500 * PBBANK= 000001 *               * STACK OP  = LEFT *
* Q       = 132200 * (P-PB)= 007147 *             * OVERFLOW  = OFF  *
* S       = 132210 *                * MAP= ON     * CARRY     = ON   *
* Z       = 136240 *                *             * COND CODE = CCE  *
*                  *                *             * SEGMENT # =  27P *
**********************************************************************
      # - SIGNIFIES THAT VALUE SHOWN IS DIFFERENT
          FROM ORIGINAL VALUE OF REGISTER.
****************************************************************************
*                             S I R = 140017                               *
****************************************************************************
* SYSTEM RESET           = ON         * NON RESPOND. DEVICE    = OFF       *
* SYSTEM CLOCK           = ON         * RUN/HALT FOR CMP       = OFF       *
* CHANNEL SERVICE REQ.   = OFF        * DISABLE ATTN. FLAG     = OFF       *
* EXTERNAL INTERRUPT     = OFF        * DATA NOT VALID ON IMB  = OFF       *
* POWER ON               = OFF        * DISPATCHER FLAG        = OFF       *
* POWER FAIL WARNING     = OFF        * ICS FLAG               = OFF       *
* INTEGER OVERFLOW       = OFF        * SPLIT STACK MODE       = OFF       *
* MEMORY PARITY ERROR    = OFF        * RUN/HALT               = HALT      *
****************************************************************************

***** FIXED LOW MEMORY *****
   (@% 0) CST  PTR       022140       (@% 5) ICS QI              050240
   (@% 1) XCST PTR       030324       (@% 6) ICS ZI              060236
   (@% 2) DST  PTR       002140       (@% 7) INTERRUPT MASK      157120
   (@% 3) NOT USED(MPEVE)000000       (@%10) DRT BANK            000000
   (@% 4) CPCB INDEX REL 000671       (@%11) DRT ADDR            000000

***** SYSGLOB (%1000)  *****
   (+%0) SGLOB-SBASE     000000       (+% 5) IOQBASE-REL         007141
   (+%1) CST BASE-REL    021140       (+% 6) SBUF-REL            177041
   (+%2) DST BASE-REL    001140       (+% 7) ICS-QI REL          047240
   (+%3) PCB BASE-REL    121700       (+%10) LPDT BASE-REL       163540
   (+%4) SWAPTAB BASE-REL134340       (+%11) SMON  BASE-REL      166700
```

This output is usually a good place to start looking at a dump. It tells you when the dump was taken, what version of MPE the dump was taken from and what type of CPU it was. This output will show you what the contents of the registers were when the memory dump was taken. It also shows you the contents of some of the more important memory locations in the system. IDAT performs some integrity checking during the production of this output, so if you see error messages relating to the contents of memory then you may have difficulty continuing with further analysis.

IDAT, when you 'Text' in a dump file, also looks for the tell-tale signs that someone "took a dump of the dump system." If you had a problem taking a memory dump, but thought you could try it again by entering DUMP RETURN twice, then this fact will be displayed by IDAT. Usually, the second dump has wiped out nearly every important piece of information about the system registers, which makes these types of memory dumps of little value.

STEP #2

Determine whether or not we have an active process:

Check the value in absolute location %4. You will find it displayed under the heading:

```
***** FIXED LOW MEMORY *****
```

In this case, we find:

```
(@% 4) CPCB INDEX REL 000671
```

If the contents of %4 are non-zero, then we *probably* have a valid active process. To be sure, check the contents of the DB BANK and S BANK registers. If they are *both* non-zero *and* equal, then we are *not* processing on the ICS (Interrupt Control Stack). As a further check, look for the status of the ICS FLAG. If it is OFF, then we are not processing on the ICS. When the system is "on the ICS," this means that there may be no active user process discernible from the contents of the system's registers.

Absolute location %4 is a PCB (Process Control Block) relative address that points to the current process' entry. We refer to each process in the system by its PIN (Process Identification Number). The PIN corresponds to the number of the entry in the PCB table. If you are running any version of MPE V/E, each entry in the PCB table contains %25 words of data. By doing some simple arithmetic, we can determine which PIN was active when the system failed.

Divide the contents of absolute location %4 by %25.

Within IDAT, you can do this easily by entering:

```
-=671/25 RETURN
```

IDAT responds with the answer of: 25. (All arithmetic is octal unless you specify decimal numbers). This is the PIN which was active when the system failed. The contents of the stack for PIN 25 is what we are now interested in, so we will proceed to Step #3.

STEP #3

Format the current PCB entry that we just found:

```
-F A4:,PCB RETURN
```

```
  PROCESS ID                                                        SCHEDULE INFO
--------------                            RESOURCES     PSEUDO INT  -------------
  PIN: % 25                    EVENT    -------------  -----------     PRI: 225
  (CURRENT)                    FLAGS        CRIT:       PSIM: NORM   WSOFT:
```

```
PTYPE: SYST                   -----      HSIR:        SI:        DISPQ: YES
 NAME:               M:                    SC:        HK:           LQ: YES
                    RG:              NEXT IMP:        SK:           CQ:
                    RL:              PREV IMP:        ST:           DQ:
DATA SEGMENTS       MA:                               HB:           EQ:
--------------     BIO:             MISCELLANEOUS     CY:        INTER:
   XDS:             IO:             -------------     BK:        CORER:
 ABS DB:          UCOP:                  BMS: SNF  RITBK:        ASOFT:
                  JUNK:                  PPC: NUL  PIOVR:        HIPRI:
  STACK: 157     TIMER:               OBJID1:                    USEDQ:
SOV ALC:           MSG:         YES  PBX PTR: 12   DSSRVR:         TRW:
                   SON:               SL PTR: 2624                  SW:
                 FATHR:               BPLINK:      LIFE/DEATH       LW:
 FAMILY INFO       IMP:                            -----------   DSOFT:
--------------     SIR:               QUEUE LINKS    LIVE: YES      PC:
  FATHER: 24     TMOUT:             -------------    DEAD:       IPEXP:
     SON:      MEM/WWS:             NQPIN:             FAC:      HSPRI:
 BROTHER:                           PQPIN: 37                      SAR:
      OA: F  PROG FILE LABEL LDEV: 1  DISC ADDR: %00000166215   P SOV:
```

We could have obtained the same output by specifying the PCB entry specifically. In this case, the following command would also have worked:

```
-F PCB25 RETURN
```

This output shows just about everything you would ever want to know about the current status of the process requested. When looking for the current process, check to make sure that the word `(CURRENT)` appears underneath the line: `PIN: %xxx`. If the designation `(CURRENT)` does not appear, then you should check to make sure that the command that you entered is correct, or that your octal arithmetic is correct.

For our purposes, we should first determine which data segment is the stack for this process. In the left column of the display there is a line which shows us the stack data segment number. In this case, it is %157. Having found what we want, we can proceed to Step #4.

STEP #4

Format the stack markers for the current process:

```
-PMAP RETURN
```

The above command, if available with your version of IDAT, can be entered when you are examining a dump which came from the same system where you are now running IDAT. This command can also be entered if you are looking at a dump that came from a version of MPE that is *exactly* the same revision as the one you are running. This command tells IDAT to look up the procedure names for the stack markers associated with MPE.

```
-F DA157,STACK RETURN


FORMATTED STACK.    DST 157

***PXGLOBAL***

013 126430: 001050  001200  177777  000000  177777  000000
013 126436: 006000  000000  000024  000024  000063  000045

SEG REL DL: 001050 SEG REL DB: 001200 JMAT INDEX:   000000 JPCNT INDEX:  177777
JOB IP LDN: 000024 JOB OP LDN: 000024 JDT DST INDX: 000063 JIT DST INDX: 000045
JOB TYPE:  UNDEF   DUP:         YES    INTERACT:     YES    JCUT INDEX:   000000

* CURRENT PROCESS *

BANK ADDR.    X    DELTA'P STATUS DELTA'Q SEGMENT          PROCEDURE       OFFSET

013 132175: 000702 053074 140052 000152 MORGUE'ABORT (15 ABORT              00522
013 132023: 032057 044731 140001 000012 ININ
013 132011: 000112 001615 142404 000025 USER SEGMENT
013 131764: 000010 001461 142004 000011 USER SEGMENT
013 131753: 000004 001403 142004 002100 USER SEGMENT
013 127653: 000000 000265 160002 000006 USER SEGMENT
013 127645: 000000 040000 140052 000004 MORGUE'ABORT (15 TERMINATE          00000
```

The output received from the this formatting command is now what we need to examine. IDAT has flagged this stack as the stack of the current process. Each stack marker for this process is then traced and displayed for us, starting with the *most recent* stack marker. Under the column labeled SEGMENT we can trace the MPE segment which was being executed, along with several 'USER SEGMENT' markers. The USER SEGMENT markers represent calls between procedures within the program itself. The rest of the segments listed are the MPE system segment stack markers. For each MPE segment, the prior PMAP command has caused IDAT to look up the name of the MPE procedure within the segment which was called. These names appear under the column labeled: PROCEDURE.

In this case, there is a segment identified as ININ in the list. This is important, because an ININ stack marker means that the system had to process an INternal INterrupt. When the system processes an internal interrupt, it keeps a record in the stack of the active process of the type of interrupt that was processed.

One guideline that I was informed of while looking at dumps is that an ININ stack marker should probably be the first stack marker that you examine. In this case, we have had a SF310. Your reference manuals will tell you that this is caused by a system process aborting. Processes can abort from a variety of causes, some of which include stack over/underflows or bounds violations. Exactly which type has occurred here is what we need to determine, so let's proceed to Step #5.

STEP #5

Display the stack in the vicinity of the ININ stack marker:

```
-D EA13+132000,40,B RETURN

013 132000 (000000): 000200 000001 000000 000001  ........ (000000)
013 132004 (000004): 000000 000305 000021 000016  ........ (000010)
013 132010 (000010): 141420 000112 001615 142404  ...J.... (000020)
013 132014 (000014): 000025 100401 032057 000671  ....4/.. (000030)
013 132020 (000020): 000400 000030 000000 032057  ......4/ (000040)
013 132024 (000024): 044731 140001 000012 000002  I....... (000050)
013 132030 (000030): 056136 141104 000010 001213  \^.D.... (000060)
013 132034 (000034): 000100 000042 110414 001213  .@.".... (000070)
```

The area of memory that you want to display is determined by the stack marker trace that we saw after the `-F DA157,STACK` command. Under the column labeled `BANK ADDR.` we see the memory locations that we will be looking at. When you look at memory around stack markers, choose enough so that you can see the data that you need to, but not so much that you have to wade through pages of listings or several screens of output. In this case, there is enough memory displayed to include the previous USER SEGMENT stack marker and several words beyond the location of the ININ stack marker.

The bank and address offset of each stack marker displayed in Step #4 is the address of the *first* word of the marker. Therefore, if we are interested in the stack marker *previous* to the ININ marker, it is at location EA13+132011 and extends for four words. They are shaded below:

```
013 132000 (000000): 000200 000001 000000 000001  ........ (000000)
013 132004 (000004): 000000 000305 000021 000016  ........ (000010)
013 132010 (000010): 141420 000112 001615 142404  ...J.... (000020)
013 132014 (000014): 000025 100401 032057 000671  ....4/.. (000030)
013 132020 (000020): 000400 000030 000000 032057  ......4/ (000040)
013 132024 (000024): 044731 140001 000012 000002  I....... (000050)
013 132030 (000030): 056136 141104 000010 001213  \^.D.... (000060)
013 132034 (000034): 000100 000042 110414 001213  .@.".... (000070)
```

Just beyond the end of this stack marker is where the ININ process laid down its information. The word laid down was the type of interrupt being processed. This is shown shaded below:

```
013 132000 (000000): 000200 000001 000000 000001  ........ (000000)
013 132004 (000004): 000000 000305 000021 000016  ........ (000010)
013 132010 (000010): 141420 000112 001615 142404  ...J.... (000020)
013 132014 (000014): 000025 100401 032057 000671  ....4/.. (000030)
013 132020 (000020): 000400 000030 000000 032057  ......4/ (000040)
013 132024 (000024): 044731 140001 000012 000002  I....... (000050)
013 132030 (000030): 056136 141104 000010 001213  \^.D.... (000060)
013 132034 (000034): 000100 000042 110414 001213  .@.".... (000070)
```

The interrupt codes can be looked up in any of the older MPE pocket guides in the ASCII/Instruction Set section. The one I happened to use listed them on page 11-14. There you can find that the interrupt code of %100401 is caused by a bounds violation. When the system detected that one of its *system processes* aborted, a call was generated to the MPE procedure SUDDENDEATH with a parameter of 310. This produced the failure message that appeared on the console.

NOTE

With more recent releases of MPE, the Internal Interrupt routine may not save a record of certain types of interrupts. Also, the older brown MPE pocket guides may not list *all* the possible interrupt codes. If you try this method of analysis on a dump of your own and you do not seem to find any meaningful information in the stack, please let the Response Center assist you in your analysis.

Now, how can we find out more specifically which process aborted? Let's go back to the formatted display of the current process' PCB entry (Step #3). At the bottom of the display is a line containing the disc address of the file label for the program which was running at the time of the failure. IDAT can sometimes look up the process' file name in memory, in which case it will be displayed in this area of the output. In this case, however, IDAT was not able to determine the process' file name, so as a next-best effort, it looked up the disc address of the program file, which was also stored away in one of the MPE tables.

Since, in our scenario, we are looking at our own dump, unless the program file was purged after the failure *and* overwritten with new data, we can use one of the other TELESUP utilities to look up this sector on the appropriate disc and display the contents of the file label. DISKED5.PUBSYS can do that for you. In this case, the program file happened to be NMLOGMON.PUBSYS.

The program NMLOGMON.PUBSYS is one of the programs run by HP communications software. This program is activated when some HP communications subsystems are brought up.

If I were still working for HP, and if I happened to be one of the "on-loan" engineers working this problem for a customer, I would now begin to start asking questions about what was going on at the time of the failure. Perhaps more importantly, I would also ask about anything that may have changed recently. Were any Communication System devices reconfigured recently? What communication products exist on the system? Were any changes made to MPE recently?

In this case, let us see what else we can determine about the status of the the machine at the time of the failure. Proceed to Step #6.

STEP #6

Determine who was signed on and what they were doing:

```
-F JOB (RETURN)

JOBNUM STATE UMAIN JIN  JLIST    INTRODUCED JOB NAME
              PIN#
#S2     EXEC   26   20  20       SAT  7:15P MANAGER.SYS,PUB

  1   SESSIONS, 0   JOBS
 JOB LIMIT = 1   SESSION LIMIT = 1
 JOBFENCE = 7
 JOBSECURITY = HIGH
```

The output generated by this command shows us who was signed on at the time of the failure, similar to what a `:SHOWJOB` would do for us while the system is up. We can see that there was only one user on the system at the time. By entering the command below we can see what this user's last command was:

```
-F CI (RETURN)

JOBNUM UMAIN STACK JOBNAME          LAST COMMAND
        PIN#  DST#
#S2      26   150  MANAGER.SYS      NETCONTROL START
```

We have now determined that the only user on the system had just entered a `:NETCONTROL START` command. Since NMLOGMON.PUB.SYS is one of the programs which would be activated during this process, it stands to reason that the problem is related to the use of the Network Services on this system.

In order to determine an exact cause of the failure at this point, it may be necessary to leave the realm of IDAT and pursue other courses of action. If I had access to the source code for the NMLOGMON.PUB.SYS program file, I could use it to trace the 'USER SEGMENT' stack markers and attempt to determine what procedures were being called, what data was passed to those procedures, etc. But since I am no longer privy to that kind of information, this would be the time to let the Response Center take over and further research the problem.

When this failure actually happened, however, I elected to do one other task while waiting on the Response Center. I used a utility called CHECKSUM to examine the NMLOGMON.PUB.SYS program file. This utility can calculate a unique checksum for a program and can verify this checksum against a checksum stored permanently as part of a program. I realize that I am digressing away from the topic of this paper, but in order to satisfy those who may be curious, I will elaborate just a little more on this example.

The CHECKSUM utility found a discrepancy between the calculated checksum and the checksum stored permanently in the program file. It happened that another Boeing site in Seattle had already updated several machines to this version of MPE, and had no problems bringing up their networks. Dialing into one of their systems, I again used the CHECKSUM utility to verify that NMLOGMON.PUB.SYS in Seattle was the same as the copy that I had. It was *not.*

Running CHECKSUM against NMLOGMON.PUB.SYS in Seattle told me that their copy was not corrupt, but mine was. Later, it was confirmed by HP that something had corrupted the file on a few of the MIT tapes that had been distributed to the field. By using DSCOPY to retrieve a *good* copy of the program, I was able to bring up the network without any problems.

What you can see from this example is that IDAT may be only one of several tools that you may have to use to determine the cause of a failure or system hang. In an actual scenario such as this SF310, however, there are still other items of interest in the contents of the memory dump that may be worth checking, just to be sure that you haven't overlooked something.

The following are examples of other IDAT commands which will aid you in determining the integrity (or lack of it) of MPE at the time of the failure. These commands are provided as examples of various commands and are not meant to imply that they are the most appropriate commands for use in analyzing a SF310.

Addenda

Check the integrity of system code segments in memory:

```
-F CKSUM RETURN

COMPUTED    RETRIEVED               SEG   SEG                 PROG
CHECKSUM    CHECKSUM     COMPARE    NUM   NAME                NAME
--------    ---------    -------    ---   -----------         --------
 127614      164241      NORMAL       1   ININ                P10P033C
 017412      017412       OK          6   CLIB'03
 074344      074344       OK         15   NLS02'3
 022724      022724       OK         16   NLS01'4
 052771      052771       OK         17   FILESYS4'6
 173125      173125       OK         20   FILESYS1'8
 024414      024414       OK         21   FILESYS7
 144235      144235       OK         22   FILESYS6A
 141476      141476       OK         23   FILESYS5
   .           .           .          .     .
   .           .           .          .     .
   .           .           .          .     .
```

```
162514      162514        OK       215  NMADSFMTSEG1
010744      010744        OK       216  NMSEGF
044646      044646        OK       220  NETUI
001123      001123        OK       221  SEG'            P43P002C
001144    *NO PATCH AREA*          222
174302      174302        OK       223  SEG'            P44P033C
121760    *NO PATCH AREA*          224
075511      075511        OK       225  IOLANO'MPE5     P21P131A
```

In one word of each MPE system code segment, the systems programmers place the checksum of the segment *as it was originally created.* IDAT finds this original checksum and also computes a new checksum using the same algorithm. This command displays the two checksums for you so that you can instantly determine if there was any corruption of memory that dealt with MPE. If you find any, then you need to determine if the copy of the corrupt segment(s) in SL.PUB.SYS is/are also corrupt. If the contents of memory are corrupt, but the copy on disc is *not*, then restarting the system will replace the corrupt segment with one that is valid. If the segment on the disc is *also* corrupt, then you will need to perform a COLDLOAD or UPDATE from a *known-good* tape. That is why you *always* need to know where you keep a valid COLDLOAD tape, since, if your disc has been corrupted, then one or more backup tapes are also probably corrupt. If you COLDLOAD or UPDATE from one of these corrupt backup tapes, you will not be free of the problem.

NOTE

The placement of a checksum in an MPE code segment or in a program file depends on the use of an undocumented option in the Segmenter which tells it to calculate a checksum and save it in the proper place. There may be one or more MPE segments which were not *PREP*ed with this option in certain versions of MPE starting with V-Mit. I was informed that one such segment is named: `ASVTSEG2`. This segment is part of the Network Services software (NS/3000). Before you panic at finding a "corrupt" segment, at least according to IDAT, check with the Response Center to see if, in fact, you *do* have something to be concerned about.

How do you tell if the segments on disc are corrupt? One way would be to use the CHECKSUM utility previously mentioned. The other way is to use IDAT in "live" mode. After you have restarted your machine, new copies of the MPE code segments will have been loaded into memory. IDAT has an option which merely switches its function from that of formatting the contents of a disc file to that of formatting the *live* system. The command is: `-L` RETURN. If we switch IDAT into "live" mode, then we can use the same `-F CKSUM` command as we used above to format what is currently in memory while the system is up and running. If IDAT shows the same corruption as was contained in the memory dump, then you will need to COLDLOAD or UPDATE from a good tape in order to remove the corruption.

Check for any privileged mode programs:

```
-F PMUSERS RETURN

 PIN %227    STK %636    OPT.PUB.SYS
 PIN %230    STK %363    QUAD.PUB.XTOOLS
```

If you see *any* entries listed, they are candidates for examination concerning the failure. Even programs which are supplied and supported by HP could possibly cause system failures under the proper circumstances.

As we found out in our first scenario, a corrupt program file was at the root of the cause for the SF310. Given other conditions, this program, which runs in privileged mode, could just as easily have modified memory in some location which would have caused a totally different system process to abort, or caused random memory corruption to other users' programs or data segments. Failures caused by this type of corruption are usually *very* difficult to trace.

Let us consider a few other integrity-checking commands within IDAT.

Verify various memory structures:

```
-F ARL RETURN

TESTING LINKS OF THE ARL. #REGIONS in LIST = 61
FINISHED CHECKING ARL. NO ERRORS FOUND.
                                MPE 5/E FORMAT
               REG.  PREVIOUS        NEXT   A R A C S L F I M  OBJ. OBJ. CSTX
 HDR ADDRESS   SIZE HDR ADDRESS HDR ADDRESS S S V L C K Z O I  TYPE NUM. INDEX
 022 177600       1 000 000000  023 055400      Y               DST 751
 023 055400       1 022 177600  027 133400      Y               DST
 027 133400       1 023 055400  024 056600      Y               DST 1110
                           .                    .                .
                           .                    .                .
                           .                    .                .
 016 144200      60 026 042400  014 135000      Y               DST
 014 135000      62 016 144200  035 000000      Y               DST 226
 035 000000      65 014 135000  024 000000      Y              CSTX 14    165
 024 000000      67 035 000000  020 155400      Y               CST 170
 020 155400      74 024 000000  034 127000      Y              CSTX 33    165
 034 127000      74 020 155400  034 047000      Y               DST 213
 034 047000      79 034 127000  022 000600      Y              CSTX 15    165
 022 000600      81 034 047000  011 043200      Y               CST 66
 011 043200      85 022 000600  015 070200      Y               CST 141
```

```
015 070200    86  011 043200  034 000000       Y               DST 221
034 000000    87  015 070200  021 146600       Y               DST 172
021 146600   101  034 000000  000 000000       Y              CSTX 27    165
```

This command traces and checks the Available Region List (ARL) of your system's memory. Note that the region sizes proceed from the smallest to the largest. IDAT performs some integrity checking as it produces the list so, for example, any backward pointer which did not point to the previous region in the list would be flagged as corrupted.

In the right-most column you will see a representation of what the segment identifier says that this segment is. In this case, the identifier for an available region is mostly meaningless. There are other memory-formatting commands which *will* display meaningful information here.

One of these commands would be:

```
-F MEM RETURN
```

This command starts at the low memory addresses and formats information contained in each memory region header and trailer. Again, IDAT performs some integrity checking along the way. This command can produce a good deal of output, especially on larger machines. Space being limited here, I have not included an output example. The format of the output is the same as that of the -F ARL command shown above, however.

If MPE disc caching was in use, check the Cache Directory Table (CDT):

```
-F CDT RETURN
```

```
Cache Directory Table Header.   (DST %206 Address %004 100230
*************************************************************
Num entries = 001611 | Entry size = 000032 | Num free = 001605
First free  = 021242 | Last free  = 021210 | Max used = 000007
Ldevs cached= 000002 | First entry= 000001 | DST size = 055752
Stop pending= 000000 | Sectors seq= 000140 | Sec. rand= 000020
Force post  = 000000 | Head imp q = 000000 |
*************************************************************

Cache Directory Table - Device Entry #1
*************************************************************
Next ldev = 000077 | Prev ldev = 000000 | This ldev = 000002
Num pages = 001121 | # domains = 000000 | Map head  = 000000
Map tail  = 000000 | # regions = 000043 |
.............................................................
Head domain pntr = 015 017623 | Tail domain pntr = 022 060623
Number read hits = 611        | Number write hits= 406
Number read miss = 47         | Number write miss= 20
Number proc stops= 147        | Scan pointer     = 002 177223
*************************************************************
```

```
Cache Directory Table - Device Entry #63
****************************************************************
Next ldev = 000000 | Prev ldev = 000001 | This ldev = 000004
Num pages = 001055 | # domains = 000000 | Map head  = 000000
Map tail  = 000000 | # regions = 000035 |
----------------------------------------------------------------
Head domain pntr = 012 162623 | Tail domain pntr = 007 137423
Number read hits = 153        | Number write hits= 61
Number read miss = 74         | Number write miss= 1
Number proc stops= 72         | Scan pointer     = 010 112623
****************************************************************
```

If MPE disc caching was turned on for any of your disc drives, then this command will tell you which devices were cached. You can then use another command to format the Cached Domain Regions for each disc. Again, integrity checking is done on the disc domain regions. If the system had failed with an error indicating some type of caching problem, then tracing the cache domain regions might be very helpful in finding the cause.

This output shows that two discs, logical device numbers two and four, respectively, were cached when the system failed. The logical device number is displayed after the heading: `This ldev =`. The heading: `Device Entry #` does *not* refer to the logical device number. This number corresponds to the entry number in the Cache Directory Table, instead.

Check the cached domains for each drive cached:

```
-F CDR 2 (RETURN)
```

```
CACHED DOMAIN REGION HEADERS FOR Ldev 2
# OF REGIONS IN THE LIST = 134
TESTING THE LINK POINTERS OF THE CDR
FINISHED CHECKED CDR. NO ERRORS FOUND.
                                 MPE 5/E FORMAT
             REG.   PREVIOUS        NEXT     A R A C S L F I M  OBJ. OBJ. CSTX
 HDR ADDRESS SIZE HDR ADDRESS  HDR ADDRESS  S S V L C K Z O I  TYPE NUM. INDEX
 010 171600     2 000 000000  025 113400  Y     Y            CDT
 025 113400     2 010 171600  114 134200  Y     Y            CDT
 114 134200     2 025 113400  075 152200  Y     Y            CDT
 075 152200     2 114 134200  133 122000  Y     Y            CDT
 133 122000     2 075 152200  055 072000  Y     Y            CDT
 055 072000     2 133 122000  137 172600  Y     Y            CDT
       .        .       .           .           .             .
       .        .       .           .           .             .
       .        .       .           .           .             .
 124 136400    17 013 064400  061 066000  Y     Y            CDT
 061 066000     2 124 136400  121 035600  Y     Y            CDT
```

```
121 035600    61  061 066000  101 145000  Y      Y              CDT
101 145000    71  121 035600  065 156200  Y      Y              CDT
065 156200    71  101 145000  125 012000  Y      Y              CDT
125 012000    17  065 156200  043 043600  Y      Y              CDT
043 043600    17  125 012000  137 034600  Y      Y              CDT
137 034600    17  043 043600  000 000000  Y      Y              CDT
```

(I'll only include one disc for this example)

When you format the Cached Domain Regions (CDR) for a particular disc drive, what you should look for at first is to make sure that no integrity errors were detected. Also, each object type appearing under the heading of `OBJ. TYPE` should be labeled a CDT. If you see items in the list which are identified as something else (CST, DST, CSTX) then you may have cause to suspect something is wrong. Let me reiterate, however, that I can only provide "rules of thumb" for most of what you might see. Don't forget that IDAT is officially an unsupported utility, even though the Computer Systems Division (CSY) at HP *does* maintain it and also develops new capabilities for IDAT. As newer versions of MPE are released, however, there may be structural changes which even recent versions of IDAT may not handle correctly. If you have any doubts at all about what IDAT tells you, please let the Response Center assist you.

Let us consider another type of failure, for example. This time, it will be a system hang. Again, this example came from an actual system hang, but the names have been changed to protect the innocent.

In order to get a valid dump of a system that is hung, you must HALT the system execution and immediately start the SDF. Most of the time, you can do this by entering Ctrl-B at the system console. When the system returns with its prompt, you would enter `HALT` RETURN. After the machine halts, you can invoke the SDF software to take the dump. If, for some reason, you cannot get the system to respond after entering a Ctrl-B at the console, then you will have to try using physical means to halt the machine. On Series 4x/5x machines, you can open the front panel of the CPU and press the button labeled HALT. On Series 6x/7x machines, and also the newer Series 37's and Micro 3000's, you may have to resort to attempting to trick the system by causing a power failure. If that procedure fails, you may be out of luck. Phone the Response Center for further assistance, in that case.

STEP #1

Format the system registers:

```
-F REG RETURN
```

```
                    *****   REGISTERS   *****

 4/01/88,  4:28PM     MPE 5 (G.A3.01) (BASE G.A3.01) * SERIES 4x/5x  *
**************************************************************************
*   DATA SEGMENT   *  CODE SEGMENT  *MISCELLANEOUS* STATUS = 100513  *
**************************************************************************
* DB BANK = 000003 * PB     = 140430 * X  = 001271 * MODE       = PRIV *
* DB      = 035030 * P      = 157365 * CIR= 000700 * INTERRUPTS= OFF  *
* S  BANK = 000000 * PL     = 173753 * NIR= 000000 * TRAPS      = OFF  *
* DL      = 177777 * PBBANK= 000002 *              * STACK OP   = LEFT *
* Q       = 050456 * (P-PB)= 016735 *              * OVERFLOW   = OFF  *
* S       = 050555 *                * MAP= ON      * CARRY      = OFF  *
* Z       = 060236 *                *              * COND CODE = CCL  *
*                  *                *              * SEGMENT # = 113P *
**************************************************************************
      # - SIGNIFIES THAT VALUE SHOWN IS DIFFERENT
          FROM ORIGINAL VALUE OF REGISTER.
*********************************************************************************
*                              S I R = 140011                                  *
*********************************************************************************
* SYSTEM RESET            = ON        * NON RESPOND. DEVICE    = OFF           *
* SYSTEM CLOCK            = ON        * RUN/HALT FOR CMP       = OFF           *
* CHANNEL SERVICE REQ.    = OFF       * DISABLE ATTN. FLAG     = OFF           *
* EXTERNAL INTERRUPT      = OFF       * DATA NOT VALID ON IMB  = OFF           *
* POWER ON                = OFF       * DISPATCHER FLAG        = OFF           *
* POWER FAIL WARNING      = OFF       * ICS FLAG               = ON            *
* INTEGER OVERFLOW        = OFF       * SPLIT STACK MODE       = ON            *
* MEMORY PARITY ERROR     = OFF       * RUN/HALT               = HALT          *
*********************************************************************************

***** FIXED LOW MEMORY *****
  (@% 0) CST  PTR        022140      (@% 5) ICS QI               050240
  (@% 1) XCST PTR        031004      (@% 6) ICS ZI               060236
  (@% 2) DST  PTR        002140      (@% 7) INTERRUPT MASK       157120
  (@% 3) NOT USED(MPEVE)000000       (@%10) DRT BANK             000000
  (@% 4) CPCB INDEX REL 007304       (@%11) DRT ADDR             000000

***** SYSGLOB (%1000)  *****
  (+%0) SGLOB-SBASE      000000      (+% 5) IOQBASE-REL          007141
  (+%1) CST BASE-REL     021140      (+% 6) SBUF-REL             177041
  (+%2) DST BASE-REL     001140      (+% 7) ICS-QI REL           047240
  (+%3) PCB BASE-REL     121600      (+%10) LPDT BASE-REL        164240
  (+%4) SWAPTAB BASE-REL134240       (+%11) SMON  BASE-REL       170000
```

Notice that we have a non-zero value in absolute location %4, but the DB BANK and S BANK values are *not* equal. Also, notice that the status of the ICS FLAG is ON. When processing certain interrupts, the system environment uses a stack area called the Interrupt Control Stack (ICS). In order to find out what is going on when we are "on the ICS," we can ask IDAT to format the ICS information for us. Continue to Step #2.

STEP #2

Format the ICS information:

```
-F ICS RETURN

          ---SCHEDULING INFORMATION---
CURR E    CURR D                        CURR C    MAX C     MIN C
FILTER    FILTER    CWTNUM    CWTDENOM  FILTER    FILTER    FILTER
001750    001750    000143    000144    000140    000144    000000

E BASE    D BASE    C BASE    E LIMIT   D LIMIT   C LIMIT
000360    000276    000230    000375    000356    000310

          ---CURRENT STACK INFO---
STACK                         TRACE
DST       PISTAT    PIADDR    FLAG      PFAILPCB  JCUT      CPCB INDX
000233    100114    044142    177777    000000    163000    007304

          LAUNCH    LAUNCH    LAUNCH              STACK
PCBX      Z         DL        S         SBANK     DB
121630    014730    000000    012627    000007    122500

          ---MISCELLANEOUS INFORMATION---
CAND      LAST      LIST      PMBC      PMBC      PMBC      PMBC      PSDB
PIN       WEIGHT    STATE     BNDS FLG  XDS SIZE  XDS BASE  XDS BANK  COUNTER
000264    000001    000000    000000    000000    000000    000000    000000

PAUSE TIME
00336335742

          ---STACK MARKERS---
ADDR.   X-REG   DELTA-P STATUS  DELTA-Q SEGMENT
------  ------  ------- ------  ------- -------
050453  000004  050435P 101113  000034  TERMDRIVER (223)
050417  000001  057062P 100071  000026  TERMHANDLR (200)
050371  000007  045121P 102071  000040  TERMHANDLR (200)
050331  000001  041575P 102113  000037  TERMDRIVER (223)
050272  000004  042715P 142113  000035  TERMDRIVER (223)
050235  000000  042350P 100111  100000  ***DISP MARKER***
```

Perhaps one of the more important things we gain from this output is an indication of which PIN was processing interrupts. This information can be found under the heading `CAND PIN` near the middle of the displayed output. Along with the PIN, its stack data segment number is also listed under the heading `STACK DST`. At the bottom of the displayed output will be any other stack marker information which is available to IDAT. Here, we can see several MPE segment names listed under the column labeled `SEGMENT`. The last stack marker displayed will always be labeled: `***DISP MARKER***`. This is a special stack marker laid down by the MPE Dispatcher. If you should find no stack markers displayed other than the dispatch marker, then you may wish to proceed to the candidate PIN's stack to see if more information can be learned there. In this case, let us do just that, even though we have other stack markers displayed here. Proceed to Step #3.

STEP #3

Format the candidate PIN's PCB entry:

```
-F PCB264 (RETURN)

     PROCESS ID                                                     SCHEDULE INFO
   --------------                        RESOURCES     PSEUDO INT   -------------
     PIN: % 264                 EVENT   -------------  -----------      PRI: 230
     (CURRENT)                  FLAGS      CRIT: YES    PSIM: BK      WSOFT:
   PTYPE: UMAIN                 -----      HSIR:          SI:         DISPQ: YES
    NAME:               M:                    SC: YES     HK:            LQ:
                       RG:              NEXT IMP:         SK:            CQ: YES
                       RL:              PREV IMP:         ST:            DQ:
   DATA SEGMENTS       MA:                                HB:            EQ:
   --------------     BIO:       YES    MISCELLANEOUS     CY:         INTER: YES
       XDS:            IO:              -------------     BK:         CORER:
    ABS DB: YES      UCOP:                  BMS: SNF   RITBK:         ASOFT:
                     JUNK:                  PPC: NUL   PIOVR:         HIPRI:
     STACK: 233     TIMER:               OBJID1:                      USEDQ:
   SOV ALC:           MSG:              PBX PTR:       DSSRVR:          TRW:
                      SON:               SL PTR: 4202                    SW:
                    FATHR:               BPLINK:       LIFE/DEATH        LW:
    FAMILY INFO       IMP:                             -----------    DSOFT:
   --------------     SIR:               QUEUE LINKS    LIVE: YES        PC:
     FATHER: 11     TMOUT:              -------------   DEAD:         IPEXP:
         SON: 213 MEM/WWS:              NQPIN: 345       FAC:         HSPRI:
    BROTHER: 310                        PQPIN: 303                      SAR:
          OA: S                                                       P SOV:
```

This output displays the status of PIN %264 on this system at the time of the failure. HP engineers and SE's commonly interchange the term PIN with PCB when referring to entries in the Process Control Block. The type of process that this PCB entry refers to is noted after the heading: `PTYPE:`. As you can see, it is specified as the type: `UMAIN`. This term stands for 'USER MAIN.' This means that this process corresponds to some user's Command Interpreter. When a user enters a command such as `:SHOWME` or `:RUN IDAT`, it is the Command Interpreter process which handles this command and performs the appropriate action necessary to complete it. In the case of a `:RUN` command, a son process to the Command Interpreter will be started, and this son process will be the program that the user has requested.

In the case of this dump, we now might be interested in which session or job corresponded to this Command Interpreter process. In order to find this out, proceed to Step #4.

STEP #4

Format the Job Master Table (JMAT):

```
-F JOB RETURN

JOBNUM STATE UMAIN JIN  JLIST    INTRODUCED JOB NAME
             PIN#
#S1817  EXEC  264   23  23       FRI 12:22P MATINFO.TBA,PUB
#S803   EXEC   60   22  22       WED  9:05A OPER.SYS,OPER
#S1877  EXEC  314   45  45       FRI  3:44P OLVA,MGR.WIPDEV,DATA
          .          .                .              .
          .          .                .              .
          .          .                .              .
#J771   EXEC  214   10S LP       FRI  3:58P CABW15,RALPH.TBD,SALLY
#S1803  EXEC  112   95  95       FRI 11:18A SCOTTY.NEAT,SCOTTY
#S1862  EXEC   21   30  30       FRI  3:07P INFO.TBA,PUB

  36  SESSIONS, 8   JOBS
 JOB LIMIT = 10  SESSION LIMIT = 50
 JOBFENCE = 0
 JOBSECURITY = LOW
```

Fortunately, the session which has PIN %264 as its `UMAIN PIN #` is the first one listed. What we can learn here is that the user's name is `MATINFO.TBA`. This user is signed on to Logical Device 23.

One item which may now be of interest is to find out what this user was doing at the time the dump was taken. Proceed to Step #5.

STEP #5

Format the Command Interpreter stacks:

```
-F CI RETURN
```

```
JOBNUM UMAIN STACK JOBNAME          LAST COMMAND
        PIN#  DST#
#S1817  264    233  MATINFO.TBA     :::::::::::::::::::::::::::::::::::::
#S803    60    602  OPER.SYS        ENDIF
#S1877  314    151V MGR.WIPDEV      RUN QUAD.PUB.XTOOLS;INFO="Q;T TBUKS
         .          .                .          .
         .          .                .          .
         .          .                .          .
#J771   214   1472V RALPH.TBD       RUN QUERY.PUB.SYS
#S1803  112    403V SCOTTY.NEAT
#S1862   21   1415V INFO.TBA        RUN TBAINFO.TBALM;LIB=P
```

The user `MATINFO.TBA` has, according to IDAT, a long string of colon characters as his last command. This is definitely not a normal occurrence. Let us remember this, but continue on with the task of gathering information.

STEP #6

Format the CI stack:

```
-F DA233,STACK RETURN

 FORMATTED STACK.    DST 233

 ***PXGLOBAL***

 007 121630: 000650  000650  000002  000001  000102  000000
 007 121636: 016000  000000  000027  000027  000402  001533

 SEG REL DL: 000650 SEG REL DB: 000650 JMAT INDEX:    000001 JPCNT INDEX:  000102
 JOB IP LDN: 000027 JOB OP LDN: 000027 JDT DST INDX: 000402 JIT DST INDX: 001533
 JOB TYPE:  SESSION DUP:         YES    INTERACT:     YES    JCUT INDEX:   000000

 * CURRENT PROCESS *

 BANK ADDR.     X   DELTA'P STATUS DELTA'Q SEGMENT          PROCEDURE       OFFSET

 007 135322: 000022 063074 142105 000033 TERMANAGER (215) IM'READ          01122
 007 135267: 000011 055652 140054 000075 TERMONITOR (161) TERM'LOG'MON     01174
 007 135172: 000001 047052 140427 000010 HARDRES (132)    AWAKEIO          00070
 007 135162: 000001 057736 141027 000032 HARDRES (132)    P'ATTACHIO       00713
 007 135130: 000137 057020 140027 000033 HARDRES (132)    ATTACHIO         00402
 007 135075: 000427 047252 140517 000174 FILESYS1A (227)  IOMOVE           03231
 007 134701: 000000 042355 142517 000116 FILESYS1A (227)  FREAD            00377
 007 134563: 007315 050603 141057 000013 UTILITY1 (164)   READ             00130
 007 134550: 165671 041315 140441 000603 CIINIT (144)     COMMANDINTERP    01271
 007 133745: 007304 043664 140114 000037 MISCSEGC'CHECK ( PSEUDOINT        00642
007 133706: 177756 070033 101111 000016 KERNELC (221)    WAIT            00655
007 133670: 005564 066765 101111 000014 KERNELC (221)    AWAKE           00272
007 133654: 000005 047174 141035 002077 CIPREPRUN (140)  CXPREPRUN       03333
007 131555: 007304 044754 140441 000602 CIINIT (144)     COMMANDINTERP   04730
007 130753: 000045 047063 140074 000261 UDC (203)        FEEDCI          00457
007 130472: 000000 040555 140074 000572 UDC (203)        UDC             00260
007 127700: 007304 044140 142041 000603 CIINIT (144)     COMMANDINTERP   04114
007 127075: 000007 047063 140074 000261 UDC (203)        FEEDCI          00457
007 126614: 000000 040555 140074 000572 UDC (203)        UDC             00260
007 126022: 000000 045610 141074 002006 UDC (203)        INITUDC         03233
007 124014: 000000 043605 140041 000605 CIINIT (144)     COMMANDINTERP   03561
007 123207: 000000 040000 140052 000004 MORGUE'ABORT (15 TERMINATE       00000
```

The *most recent* stack marker shows us executing in the MPE segment named TERMANAGER. Considering that the formatted stack markers on the ICS were TERMDRIVER and TERMHANDLR, it may be assumed that the system was trying to process some sort of I/O for this user. Even knowing this much can help us perform some detective work before the Response Center returns our call. It would probably be a good idea to find out who was actually using logical device 23 at the time the system hung. Once found, you could ask him/her a few questions about anything that s/he may have seen on the terminal that was out of the ordinary. There might have been some sort of application error message that appeared, or even some sort of MPE error message. Finding out as much as you can before HP begins their analysis will help speed the dump analysis along, in most cases. For this particular dump, there were thirty-six sessions signed on when the system hung. Knowing which user you should contact first is obviously a time-saver.

As with the first failure example, we can also perform some preliminary checks of MPE integrity. Some possible areas for exploration are shown below.

Addenda

Check for possible causes of deadlocks:

```
-F SIR (RETURN)

 FORMATTING...

 SIR # 45 LOCKED BY PIN #251              FILE INTEGRITY
 NO IMPEDED PROCESSES

 NO DEADLOCKS FOUND
```

This command checks the status of the SIR table. It will display the name of any SIR that was locked at the time of the failure and also display the PIN number of the locking process. Subsequently, it checks the SIR table for any deadlocks. If we had found a deadlock situation for this example memory dump, it might have meant that something else was at the root of the system hang.

NOTE

A deadlock caused by two processes in a SIR lock is not the only way that a system can become hung. File locking problems, known as RIN locks, and IMAGE/TurboIMAGE locking problems can also cause a system to hang. IMAGE/TurboIMAGE locking problems usually have to be tracked down "manually," i.e., IDAT does not have a special command to track down and display an IMAGE/TurboIMAGE deadlock.

Format the RIN table:

```
·F RIN (RETURN)

Total number of RINs configured:        1024
Total number of GLOBAL RINs configured: 64

Current number of LOCAL RINs in use:  63
Current number of GLOBAL RINs in use: 1
Current number of FILE RINs in use:   67
Current number of RINs AVAILABLE:     893

GLOBAL RIN information:

RIN#    CREATOR               PASSWORD
====    =======               ========
3       WP      .HPOFFICE     WORDRIN

ACTIVE RINs STATUS:

RIN# 1      FILE    Holding PIN %0     Waiting PIN %0     NOT LOCKED
RIN# 2      FILE    Holding PIN %0     Waiting PIN %0     NOT LOCKED
RIN# 3     GLOBAL   Holding PIN %0     Waiting PIN %0     NOT LOCKED
RIN# 5     LOCAL    Holding PIN %0     Waiting PIN %0     NOT LOCKED
RIN# 7     LOCAL    Holding PIN %0     Waiting PIN %0     NOT LOCKED
RIN# 8     LOCAL    Holding PIN %647   Waiting PIN %0     LOCKED, NO WAITING
RIN# 10     FILE    Holding PIN %0     Waiting PIN %0     NOT LOCKED
RIN# 11    LOCAL    Holding PIN %0     Waiting PIN %0     NOT LOCKED
RIN# 12     FILE    Holding PIN %0     Waiting PIN %0     NOT LOCKED
RIN# 13    LOCAL    Holding PIN %0     Waiting PIN %0     NOT LOCKED
             .            .                  .                  .
             .            .                  .                  .
             .            .                  .                  .
RIN# 180    FILE    Holding PIN %0     Waiting PIN %0     NOT LOCKED
RIN# 181   LOCAL    Holding PIN %0     Waiting PIN %0     NOT LOCKED
```

```
RIN# 182    FILE    Holding PIN %0    Waiting PIN %0    NOT LOCKED
RIN# 183    FILE    Holding PIN %0    Waiting PIN %0    NOT LOCKED
```

This command displays the status of the RIN table, listing RIN numbers, type of RIN (local or file), holding PIN (if there is one), waiting PIN (if there is one) and lock status. What you should look for, if you wish to research possible deadlock situations, is for a locked RIN *with* one or more other PINs waiting. Look for a chain of waiting PINs. If one is found, then further research may be necessary to determine if the system hang occurred because of this situation.

Check the integrity of MPE system code segments:

```
-F CKSUM [RETURN]
```

You should examine this output for any segment corruption. If any is found, it should be noted and this information should be made known to the Response Center engineer who may be helping you.

Check for any privileged mode programs:

```
-F PMUSERS [RETURN]
```

Make note of any user-written programs, TELESUP utilities and/or third-party software. Remember that a malfunctioning privileged-mode program can wreak havoc on your system.

Format the Monitor Table:

```
-F MON [RETURN]

                   ******    MONITOR TABLE    ******
LOC     PIN   EVENT
171550    264 SIODM-EXIT        000002 000000 060013 000000 000000 147131
171540    264 SPECIALRQ         000000 000731 000003 000000 000000 000000
171530    264 SIODM-ENTRY       000002 033122 040213 000013 000000 147130
171520    264 GIPINTERUPT       000002 033122 040413 000017 000000 147127
171510    270 QUIESCE           000000 000000 110356 000000 000000 147114
171500    270 SIODM-EXIT        000004 000000 060013 000000 000000 147106
171470    270 SPECIALRQ         000000 000314 000003 000000 000000 000000
171460    270 SIODM-ENTRY       000004 006531 040213 000013 000000 147104
171450    270 GIPINTERUPT       000004 006531 040413 000017 000000 147104
171440    117 QUIESCE           020040 004000 110356 000000 000000 147073
171430    117 SIODM-EXIT        000002 033122 060400 000013 000000 147072
171420    117 STARTIO           102461 000002 000132 133122 060000 147072
171410    117 SPECIALRQ         000000 000731 000000 000001 000000 000000
171400    117 SIODM-ENTRY       000002 033122 040000 000000 000000 147070
171370    271 QUIESCE           000140 000200 110676 000000 000000 147041
171360      0 SWAPIN            000271 140004 000000 000000 000000 000000
171350      0 FETCHSEG          000000 001565 000271 000000 000000 000000
171340      0 DEALLOCM          000000 000026 126030 000000 000000 000000
171330      0 FETCHSEG          000000 000666 000271 000001 000000 000000
171320    352 QUIESCE           020010 002000 122272 000000 000000 147027
171310    352 SIODM-EXIT        000004 006531 060400 000013 000000 147025
171300    352 STARTIO           101235 000004 000113 106531 060000 147025
171270    352 SPECIALRQ         000000 000314 000000 000001 000000 000000
171260    352 SIODM-ENTRY       000004 006531 040000 000000 000000 147023
171250    264 QUIESCE           020040 004000 122230 000000 000000 147010
171240    352 QUIESCE           020010 000000 122272 000000 000000 146770
171230    352 SIODM-EXIT        000001 000000 060013 000000 000000 146770
171220    352 SPECIALRQ         000000 000135 000003 000000 000000 000000
171210    352 SIODM-ENTRY       000001 035474 040213 000013 000000 146766
171200    352 GIPINTERUPT       000001 035474 040413 000017 000000 146766
171170    264 QUIESCE           030040 004000 122230 000000 000000 146763
171160    264 SIODM-EXIT        000001 035474 060400 000013 000000 146762
171150    264 STARTIO           102155 000001 000131 135474 060000 146762
171140    264 SPECIALRQ         000000 000135 000000 000001 000000 000000
171130    264 SIODM-ENTRY       000001 035474 040000 000000 000000 146761
            .                       .                    .
            .                       .                    .
            .                       .                    .
171670    352 SPECIALRQ         000000 000135 000003 000000 000000 000000
171660    352 SIODM-ENTRY       000001 002310 040213 000013 000000 146631
171650    352 GIPINTERUPT       000001 002310 040413 000017 000000 146630
```

```
171640   264 QUIESCE          030120 004000 122230 000000 000000 146625
171630   264 SIODM-EXIT       000001 002310 060400 000013 000000 146625
171620   264 STARTIO          102155 000001 000131 102310 060000 146625
171610   264 SPECIALRQ        000000 000135 000000 000001 000000 000000
171600   264 SIODM-ENTRY      000001 002310 040000 000000 000000 146612
171570    15 QUIESCE          000000 000400 140062 000000 000000 146567
```

Certain events which occur during the execution of MPE are logged in a table referred to as the Monitor table. Sometimes, formatting this table after a system hang can give you an idea of what past events were happening on the system just prior to the hang. In this case, there are many entries for PINs other than our candidate process (PIN %264). This would seem to indicate that other processes were getting some chances to perform work. If the Monitor table were filled mostly with entries that belonged to PIN %264, then this would be an indication that this was the last process that had a *chance* to run just prior to the system hang, and it may have been caught in some sort of loop at a high priority. Or, it could mean that the process was waiting for the completion of some system-related event, which then could also have been impeding other processes.

At this point, we can surmise that the system was attempting to perform I/O to logical device 23, and seemed to be having some sort of problem. Again, however, there is perhaps more information that should be gathered regarding this logical device. Does this logical device connect to the system via an ADCC or an ATP? Is there anything else about the connection of this device that is unusual or non-standard?

A listing produced by the SYSINFO utility would be able to show us what type of hardware this logical device uses. IDAT also has a command which will display configuration information about the system. To answer other questions about the connection of this device to the system, however, you may have to rely on a visual inspection. At some sites, as it is at ours, there may even be a separate department and staff that maintains the datacomm/network hardware. For the moment, let us continue with our information gathering concerning logical device 23.

Display the configuration of logical device #23:

```
-F CONFIG,23 RETURN

LDEV DRT  UNIT TYPE SUB PIN STATE CPVAO   PLABEL        CLASS     DSTATE
--------------------------------------------------------------------------
 23  11    0    16   0  264 Owned 000000  TERMONITOR    ADCC        00
```

This output shows us that, indeed, PIN %264 was the "owner" of this device at the time of the hang. It also shows us that the device is connected to the system via an ADCC. In this case, that determination was easy since a wise system manager configured the port with the device class: `ADCC`. At most sites, this would probably not be the case, so you would have to know how to tell the difference between an ADCC and an ATP with other information that IDAT would provide. Or, probably the most common utility that could be used for this task is SYSINFO. If the driver name for logical device 23 is named HIOTERM2, then the device is an ADCC; if the driver name is HIOTERM1, then the device is an ATP.

The fact that logical device 23 is on an ADCC is interesting because of the way that an ADCC handles terminal I/O. When an ADCC is "interrupted" in order to do I/O, or even perform handshaking, then the CPU is also interrupted. This differs from an ATP, where the processor board on the ATP can handle I/O "interrupts" without disturbing the processing of the CPU. In the case of this dump, it was necessary to also inform the Response Center that the actual terminal was connected through a combination of broadband network and MUX devices. Physically, it was located about two miles from the CPU.

This fact helped the Response Center determine that the ADCC for this device was, in fact, processing handshake characters when the system hung. It seemed, however, to be processing so many of them that the CPU was in a constant state of handling I/O to this device. At other user group meetings I have heard discussions about how a modem or other datacomm device that "strobes" can actually interrupt the ADCC so often as to tie up the entire system. According to the final analysis by the Response Center, that is what seemed to be happening in this example.

Example Summary

These two brief examples are intended to show you that you can gain valuable information from looking at your own memory dumps, even if you are not as trained as a Response Center engineer might be. If this discussion has stirred your interest in doing some rudimentary dump analysis for yourself, then this paper has been successful. If there are those who still want more than what was covered here, then you may have to seek out some individuals who have the knowledge *and the time* to give you further assistance. Ask around at the next user's group meeting that you have a chance to attend. Find out if there are knowledgeable people in your area who would be willing to share their skills with you.

Occasionally, HP also provides an opportunity for customers to attend a special "internals" class. I have been informed that one such class will be held in mid-1988 in California. The instructor is a gentleman from a third-party software vendor who was brought in especially for this class. If you would like to find out more information about the availability of this class, you should probably contact your local HP SE, at least as a starting point. S/he may be able to find out when and/or if the next class will be conducted.

Be aware, however, that no matter how good you may become at finding out information from a memory dump, there are many occasions when the Response Center engineers must consult the *compiled source of MPE* in order to determine the cause of some failures. Unless you also happen to have purchased the source for MPE, *and* it conforms to the version that you are running, you will probably also find that there will be a limit to how much dump analysis you can do on your own.

More about "Live" mode

IDAT can perform some functions that are similar to some other common utilities, both unsupported and supported. What follows are a few examples of useful information that IDAT can display about your system while it is up and running.

As was seen before, IDAT can format the contents of the JMAT (Job Master Table) in much the same manner as that of a `:SHOWJOB`. It can also format the contents of the Command Interpreter stacks, which shows us the last command that a job or session issued. Seeing the last command that was entered by a user is a capability that other utilities such as OPT/3000 can do. IDAT can also display information about a process that is even more technical than the information in OPT/3000. IDAT can also format information about the SIR table and RIN table, similar to OPT/3000 and other utilities.

Calling IDAT a poor-man's OPT/3000 might be stretching things a little, but there are some interesting things that you can see with IDAT in "live" mode. There are other tables whose information can be displayed in live mode, including the Timer Request List (TRL). Also, you can format the PCBX area of a stack. This will include the PXGLOBAL, PXFIXED and PXFILE area of a stack. Learning to decipher this information can lead to a determination as to what files are in use by a process and what their current status is.

IDAT also has the ability to format the Global Available File Table (GAFT), which is where entries are kept for files that are being shared by more than one process. The Segment Locality List (SLL) is another table which can be formatted.

Teaching you how to decipher each of these items is probably beyond the scope and intent of this paper. Consulting a Tables Manual is perhaps a good place to start. Or, you can also do some experimenting in order to become familiar with the information that IDAT presents. *User beware,* however. IDAT's only task that requires privileged mode while reading a memory dump on disc is that of being able to call the procedure DEBUGUTIL in order to decipher words into assembler instructions. In "live" mode, it needs privileged mode in order to read some of the areas of memory that are outside of its own stack. Be careful of older versions of IDAT while using "live" mode. Some early versions *could* cause problems for you, if you, for example, tried to display a word of memory that is outside the range of memory on your system.

In addition, if you are using IDAT in "live" mode, it is possible to format a table and receive some strange output. This is usually because you have chosen to format something which is dynamic in nature. This will not normally cause any problems to the system. If it does, however, your only alternative is to note to yourself not to do that again.

If you have any problems with IDAT malfunctioning, contact your local HP SE or the Response Center. Since IDAT is unsupported, however, HP will only attend to fixing reported problems at a low priority. At last report, the latest version of IDAT is being distributed on the same tape as the TELESUP utilities. IDAT also is one of the utilities that is included with each new version of MPE. Wherever you find it, or however you obtain it, you *can* learn to use this tool and perform some useful, interesting and helpful tasks.

IDAT Development History

The following is a brief attempt to reconstruct some of the history of IDAT as it was developed from 1982 until today. This information is presented in chronological order, to the best of my recollection. At this time, I would also like to give credit to many other individuals who have contributed or are, even now, still contributing to the success of IDAT as a dump analysis tool. Apologies are also extended to those whom I do not include in the list. Their omission is not intentional, but is more likely the result of an imperfect memory or of my own poor documentation. Thanks go to: Brian Di Silvestro, Costa Hasapopoulos and Mark Cousins, Santa Clara Response Center; Mark Hatfield, HP CEC, Mountain View, CA; Simon Cutting and Donna Gracyk, HP CSY; Marie Weston, HP Roseville, CA; Nelson Hall, HP ITG; Craig Myles, Ron Helms and Bernie Staley, HP St. Louis, MO; Jim Inghram, HP Cedar Rapids, IA (my district manager who allowed me to spend some time on IDAT development); Bruce Hemminger, Suresh Ganu and Mary Hindman, HP Cedar Rapids, IA (who put up with my ability to crash the office system in truly spectacular ways); and, of course, Bob Mead, now a manager at HP Labs, Cupertino, CA, who had the original idea for IDAT.

1982	Bob Mead develops the first pass at IDAT. Its capabilities include being able to display memory in a DEBUG-like format, along with being able to format a PCB entry and a stack.
1983	New programmers in the lab group add small amounts of functionality, such as formatting the SIR table. Staff limitations do not allow a great deal of development. Problem analysis still depends heavily on the use of DPAN.
Feb. 1984	Work starts on a separate version to handle MPE V/E memory dumps. Minor bug fixes are incorporated.
Mar. 1984	The routine to format the SIR table is rewritten, saving stack space. Use of the help file is made easier. Command parser is modified to be more user-friendly.
Jun. 1984	Octal display of memory is included when decoding machine instructions.
Aug. 1985	My first contact with IDAT while at HP. I use it to assist dump analysis occasionally. A few other SE's in the Midwest also become aware of IDAT.
Sep. 1985	I acquire a copy of IDAT which handles MPE V/E dumps, but find that supporting customers who are still on MPE IV can get confusing. Several SE's in the Midwest begin contributing suggestions for improving IDAT so that remote support can be performed quicker.
Dec. 1985	I begin toying with a copy of IDAT source. I find it to be the preferred tool for analyzing dumps, in many cases. Forcing a customer to box up a tape and/or a DPAN listing seems to be much too time consuming.
May 1986	U-Mit enhancements to SDF allow virtual memory to be copied to tape. IDAT is also modified to take advantage of this feature. At

	last, we hardly ever find a critical process' stack unreadable because it was swapped out to disc when the system failed.
Jun. 1986	Other routines from Europe are found and incorporated which format certain I/O and file information. CSY in California has again committed some staff development time to IDAT, in view of its increasing importance to the Response Centers.
Aug. 1986	Added the `-F LMAP` command, which can rebuild from the contents of memory the same information about segments that would normally be found in LOADMAP.PUB.SYS. `REDO` capability added. Added the `-F MEM` command. Fixed a bug in the routine that checks for SIR deadlocks. Fixed the formatting of the current process' stack information for certain states.
Sep. 1986	Minor clean-up done for some formatted output. Added the `-F CKSUM` command. Added data integrity checks to some display routines.
Oct. 1986	Added the `-F TRL` command. Expanded the information displayed from formatting the ICS. Added formatting commands to handle disc caching tables. Added the capability to detect the Mirco-3000 CPU types. Added the `-F CONFIG` and `-F ARL` commands.
Nov. 1986	Modified the routines for caching to handle UB-Mit CDT changes. Fixed minor bugs in formatting banks of memory.

At this point, I began working for Boeing Computer Services, but the development of additional IDAT features has continued. As this paper is being written, another enhancement to IDAT has been released: a macro facility has been incorporated into the command parser.

Hopefully, you have found this information useful, or at least interesting. IDAT will probably continue to be a very heavily used utility when it comes to analyzing dumps from any MPE-based system.

TELEPHONE CALL ACCOUNTING

Paul A. McArdle
TeleMar, Inc.
559 The Knolls of Ramapo
Suffern, New York 10901

ABSTRACT: In most organizations, telephoning represents one of the largest single expenses. Yet management of the telephone is usually a neglected function, and is often assigned as a part-time task to someone whose major interest concerns other matters.

Telephone Call Accounting, the computerized allocation and reporting of telephone usage, enables organizations to dramatically reduce telephone expenses, with initial savings of thirty percent or more a common experience.

The application of the computer to telephone management offers the Information Manager an opportunity to contribute in a major way to his or her company's bottom line and opens another door to the technology of the Office of the Future.

This paper begins with an historical background to the current environment and an overview of business telephone systems for the uninitiated. It describes the source and nature of telephone usage information, explains telephone call accounting concepts and techniques, and shows how telephone usage costs can be reduced and controlled.

HISTORICAL BACKGROUND. Management of the telephone has not until recently been a priority item with most organizations. In contrast with the extensive attention that is usually paid to data processing, with its well trained managers and technicians, telecommunications is usually left as a part-time task for the Facilities Manager, the Office Services Administrator, or the Controller. These individuals often perform merely a caretaker function. The reason for this can be found in the history of the telephone company.

Alexander Graham Bell patented his invention in 1876 and thus obtained a monopoly that lasted into the 1890's. During that time the Bell Telephone Company sold telephones and built exchange facilites in major cities. When the original patents began to run out, independent telephone companies arose to compete with Bell and much of the early innovation toward automation came from these competing firms. But by then Bell had become strong enough to limit competition by refusing the use of its intercity facilites and by absorbing many of the new independents.

In 1913 the Bell Company agreed, in what came to be called the Kingsbury Commitment, to make its intercity facilites available to independent companies, and to cease aquiring interest in any competing company unless required to do so by a regulatory agency; although, it was not until the early 1920's that real peace was achieved and these commitments kept.

During this period Government regulation of the communications industry began to develop at the state level, starting in 1907 with New York and Wisconsin. And in 1934 the Federal Communications Act provided for strict supervision of interstate operations. For decades after, telephone subscribers had to obtain all of their telephone equipment and calling facilites from the one telephone company that had a regulated monopoly in their area.

This started to change with the so called "Hush-a-Phone" court decision in 1956, which allowed physical attachment of equipment that was not manufactured or authorized by Bell. But this decree still did not allow direct electrical connection of equipment to telephone lines, only connection by electrical induction. Then in 1968, in the Carterfone case, direct connection was permitted, and so the subscriber no longer had to lease equipment solely from the phone company.

Soon after that the Federal Communications Commission, under what was termed the MCI decision, approved a new category of Specialized Common Carrier, which could compete with Bell's intercity toll facilites. Several companies began to develop competing intercity transmission facilities and other companies started renting transmission facilities in bulk from Bell and reselling them to individual subscribers at a discount from Bell's own retail price.

These two developments, the permission to connect non-standard equipment and the approval of competing common carriers, have stimulated competition and created an increasing number of alternatives for the telephone subscriber, particularly since the mid 1970's. And now the divestiture of the Bell Operating Companies by American Telephone & Telephone Company--the successor of the original Bell Telephone Company--has even further complicated the telecommunications environment.

North American telephone companies have provided the best communications system in the world, and done so with a level of service that has encouraged all but the largest organizations to rely almost completely on the telephone company for decisions relating to the kind and amount of facilites that are needed. With over half a century of experience dealing with one telephone company, most businesses are still not geared to paying much attention to the telephone bill, much less the alternatives that have become available, and executives feel ill equiped to deal with the arcane details of telecommunications.

WHY CALL ACCOUNTING?. Many executives are not convinced that the time and effort apparently needed to learn to manage their telephones will be worth while, particularly if the telephones seem to be working properly. However, there is a great deal of money to be saved. So much in fact that an investment in telephone management may pay greater dividends, particularly initially, than anything else that can be done anywhere else in the organization, including further investment in its own operation.

In organizations that permit uncontrolled use of the telephone, twenty to thirty percent of the time spent on the telephone, and a corresponding twenty to thirty percent of the monthly calling costs, is likely to be unnecessary and can be eliminated without decreasing the effectiveness and productivity of its employees. To the extent that telephoning is of a non-business nature, reduction of such phoning may even increase productivity.

This saving of twenty to thirty percent (some experience even forty percent or higher), contributes directly to the bottom line for profit making companies. When these savings, which translate directly into profits, are compared with the amount of sales that would be required to generate an equivalent profit, the significance of these savings becomes even easier to see.

In order to achieve these savings, management must be able to identify who is making or receiving which telephone calls. This information should be made available to each department head, who is perhaps in the best position to determine, employee by employee, what level of calling is appropriate. And by distributing this information on a regular basis in the form of a "telephone bill" to each individual employee, along with guidance in the expected use of the telephone, employees will become conscious and self-discipled in its use.

Furthermore, if monthly telephone expenses are allocated to each manager proportionately to the actual telephone expense incurred by his or her area, department managers are even more motivated to follow through in supervising their employees' telephone usage.

Obtaining and processing this telephone usage information is not a difficult or expensive proposition, particularly in view of the pay back.

Fundamentally there are two call accounting processing functions that must be performed: (1) determine the price of each call and (2) identify who is to pay for it. There is additional detail information, such as the date and time of the call, and what city or country it was made to, that is also useful, but call pricing and call allocation are the keys to cost control

We will next take a brief look at how the public telephone network is arranged with respect to call pricing, and then how modern business telephone systems function, as it pertains to call identification and pricing.

THE PUBLIC TELEPHONE NETWORK AND CALL PRICING. Each organization's telephone system, like its residential counterpart, is connected to a local telephone company "central office". Each central office acts as a hub for telephones that are in its immediate vicinity and that share the same central office number, which is the first three digits of the company's seven digit telephone number. Calls made from an organization to other telephones tied to the same central office are directly connected by the central office. For example, calls from 212-678-1100 to 212-678-2104 will be made through the 678 central office to which both telephones are connected.

Calls made to telephones whose hub is a different central office are connected by the telephone company via a circuit between the central offices, if the two central offices are near each other, or via one or more intermediate switching centers. The particular combination of switching centers that will be chosen may vary from call to call, depending on the number, source, and destination of other calls being made by other parties at about the same time, but the price of the call will be determined in part, not by which path was used in connecting the call, but rather the geographical relationship of the called and calling central offices.

North American telephone central offices have been grouped, as is commonly known, into areas, each identified by a three digit area code. Since the recent divestiture of the local Bell Operating Companies from AT&T, there has been an additional grouping to facilitate separate pricing by the local and long distance carriers. These new groupings sometimes fall along area code boundaries, but often they do not. These are called the Local Access and Transport Areas (LATA). Those calls contained within the same LATA are charged by the local company, while those that are made from within one LATA to telephones located in another LATA, and are thus carried by an intra-state or inter-state long distance carrier--such as AT&T, MCI, or US Sprint--are paid to that carrier.

Within each LATA there is a further subdivision which identifies local calling areas in the immediate vicinity of each central office and consist of the central offices located roughly in concentric bands around each central office. Local calls, which use to be "free" (that is, included in the basic monthly charge for the telephone service), are increasingly being charged for separately, just like long distance calls.

All of the pricing associated with telephone calls, whether they be local or long distance, are determined by government regulatory agencies at the state and federal level and published as part of regulations called "tariffs." These tariffs are rarely written by the agencies themselves; rather, nearly all tariffs are proposed and written by the telephone companies and common carriers, subject to the approval of the relevant agency.

The tariffs are arcane documents that are difficult to read and the pricing schedules are varied. Since the advent of competition in long distance service, the pricing schedules have become even more varied and complicated. However, for call accounting purposes we will see that call pricing can be simplified into two broad categories, discrete (individual) call pricing and bulk rate pricing. Either category of pricing may be distance sensitive.

Distance sensitive rates, as the name implies, vary with the distance between the calling and called parties. In regular Direct Distance Dialing (DDD), for example, calls are individually priced in this way. The distance between the originating and terminating central offices is calculated from a formula which uses the geographical coordinates from a standard set of vertical and horizontal coordinates that have been established for each central office in the U.S. and Canada.

A distance sensitive pricing schedule will list the rate for each of a set of distance ranges (e.g., 1 to 10 miles might be 25 cents for the first miniute of calling time, 11 to 22 miles might cost 27 cents, etc.). And usually there are either separate prices for day, evening, and night time calling, as well as weekend and holiday, or a basic pricing schedule is accompanied by a schedule of percentage discounts for off hours calling. Sometimes the distances are bands of states instead of miles. For example, calls to Hawaii are priced at rates that vary according the state from which the call is made (e.g., 69 cents from California, but 74 cents from New York). Foreign call pricing varies according to the country or region of the world called (e.g., a call to Sweden might cost 1.94 per minute, while one to Argentina might be 2.60).

Bulk rate pricing involves a standard rate for all calls made to certain regions of the country, or world. Rather than just distance sensitive, these prices are volume sensitive as well. The more you use the facility during a particular billing period the lower the monthly unit cost. WATS (Wide Area Telecommunications Service), for example, is an example of a bulk priced facility.

The calls that are made under a bulk pricing plan are completed through the same central office and regional switching center network as DDD calls; only the pricing is different. With volume pricing, a call made early in the billing period is likely to be more expensive than one made later in the period over the same facility, when lower rates have kicked in because of the accumulated volume of calls made earlier in the month.

For call accounting purposes, however, when pricing individual calls, we do not need to be concerned with this variation. We are only concerned with applying the average cost per minute of that facility to each call. In fact, to use the volume variation in billing rate in our call pricing may confuse telephone users, who are likely to question different costs for similar calls made at different times of the month. For organizations with a fairly consistent call volume, this average cost will be very stable, varying less than a cent per minute of calling time from month to month.

Let us turn inward now to see how business telephone calling is handled at the customer site and how it is connected to the public network.

BUSINESS TELEPHONE SYSTEMS. Very small businesses subscribe for one or more telephone business lines, which are connected to individual telephone instruments, and function just like residential telephones, although service to businesses is usually more expensive. Larger businesses will have more business lines and these may be connected to a hardware device called a Key System, which provides for an operator console, internal extension to extension calling, and other features. Each call is still initated or received on one of the individual business lines, which is connected to each individual instrument and which lights up a key on the instrument when the line is in use. Key Systems are usually limited to environments with less than one hundred extensions.

Larger companies use a system called a Private Automatic Branch Exchange (PABX) or simply Private Branch Exchange (PBX), which routes calls internally between individual users, and between individuals and the telephone company central office for outside calls, in a manner similar to the switching performed at the central office itself. Modern PBX systems, which some manufacturers choose to call by different acronyms, like IBX (Integrated Business Exchange) or CBX (Computerized Branch Exchange), are stored program digital computers which provide a host of programmable features in addition to the basic switching function.

The PBX is connected to the central office through groups of trunk lines. Each group of one or more trunks connects the organization's telephones to a particular facility at the central office. For example, there will be a group of trunks for local and long distant direct dial (DDD) calling. If the company subscribes to one or more WATS pricing plans, each of the these plans will also be represented by a group of trunks. The decision whether to make the call on DDD or on a particular WATS facility, for example, is made at the customer site, and then one of the appropriate trunks, if one in the set of trunks for that facility is currently available for use, is chosen accordingly.

Likewise, if more than one long distance carrier is subscribed to, the additional carriers may each have their own group of trunks. Normally all of these groups of trunks will go to the local central office, but will then be connected to the relevant carrier or facility. The public network is currently in the process of being converted to "equal access", which allows the user to dynamically select his long distance carrier for each particular call by dialing a five digit prefix to denote the desired carrier, if it is different from the one that had been chosen as the default carrier. This reduces the need for additional trunk groups.

Before the advent of the Automatic Route Selection feature in PBX's, it was necessary for the caller to manually select the carrier or pricing schedule (e.g., DDD vs one of the WATS facilities) that he wanted to use for the call by keying in a one or two digit prefex to tell the PBX which trunk group from which to select a trunk. This required that each employee know the best route to use, or at least follow a set of rules to determine what trunk group access code he should use according to where the call was going, and even sometimes what time of the day the call was being made. These rules were nearly impossible to enforce without call accounting, which could track incorrectly chosen routes, and even then were never fully successful.

Most modern PBX's now have Automatic Route Selection, which can be programmed by the PBX installer to choose a trunk from the least costly route of those available or, where equal access is available, to insert an appropriate prefix to access an appropriate alternate carrier. Now the business caller only has to enter one code, such as the number 9, to get an outside line, followed by the number called, and the PBX will determine the appropriate trunk group to use.

We have seen the various ways the telephone caller is connected to the public network. The specific facilites that are made availble to the individual user can be adjusted within the PBX, extension by extension, on a need to have basis. Telephones in "public" areas, such as conference rooms, lobbies, and on the production floor, might be programmed in the PBX to be limited to local calling, or even strictly for internal calling. Where outside calling is permitted, certain extensions might be programmed in the PBX to permit long distance calls, but only if a least cost facility, such as WATS, is available, while other telephones, such as those of senior executives, will be given unlimited access to outside lines.

Most advanced PBX's permit the assignment of individual telephone user Authorization Codes, each with its own set of capabilites, such that when the authorization code is entered, its capabilities will override any limitations that had been placed on that particular extension. Thus, for example, an individual with an appropriate Authorization Code might make a long distance call from a phone that is otherwise restricted to making only local calls.

Now we look at how infomation about each call is recorded and made available in computer usable form.

RECORDING THE TELEPHONE CALL. Virtually all PBX's, and even some Key Systems, are capable of recording the details of each outgoing telephone call; that is, those made from the PBX to the central office or onto a tie line connecting the PBX to another PBX. Most will also record incoming calls. Extension to extension calls, however, are normally not capable of being recorded. The contents of the computer records vary from one PBX manufacturer to the next, but typically they will contain the following information:

Date of call
Time call started or finished
Duration of call
Extension that initiated or received the call
Dialed number, if the call is outgoing
Number of the Trunk on which the call was carried
    or the Number of the Trunk Group
Authorization Code, if used
Account Code, if used

The Account Code feature permits the caller, by entering an account code at the time the call is made, to charge the call to an account other than one normally assigned to that extension or authorization code. This can be used to automate the billing of telephone expenses to clients or to project expense accounts.

Allocation of the outgoing calls to the department and the individual to be charged for the call can be based on the extension, the authorization code, or the account code.

Incoming calls can similarly be allocated based on extension or account code. The costs of incoming WATS calls, which are paid for by the receiving company, can thus be allocated to the party receiving the call. Even incoming calls that are paid for by the outside calling party should be tracked, since they represent employee telephone usage.

Notice that one crucial item of data is missing, the price of the call. This must be calculated using the other information given in the call record, in conjunction with the relevant tariff or, in the case of bulk priced facilites like WATS, the unit average cost experienced on that facility.

Price calculations cannot be one hundred percent accurate, because the recorded duration of the call is from the time a trunk was seized, or some delayed time relative to trunk seizure time, while the telephone company begins its recording of call duration for charging purposes at the time the call is answered by the called party. The telephone company central office equipment has this call start time--it is called "call supervision"--and, if the telephone companies were willing to do so, it could made it available to the business subscriber. Some PBX's have even been designed to be able to use it when and if it becomes available. Until recently telephone companies have given no indication that they will make it available to their subscribers, but there have been indications that that will change in the forseeable future.

In the mean time, however, the call accounting system must deduct an estimated setup time from the measured call duration of outgoing calls. This is normally one half minute, but it can be greater in areas where the central office switching equipment is slow. It is possible to tune this time by comparing the number and duration of calls as recorded by the PBX during a given billing period with the number and duration of calls actually being billed by the telephone company.

In the case of calls subject to discrete pricing, such as DDD calls, the price calculation can be one hundred percent accurate as far as the tariff is concerned, but may vary from the actual price charged by the carrier because the estimated call duration may differ with the duration used by the phone company.

Once calls have been allocated and priced, the call detail information is ready for summarization and reporting.

USING THE CALL INFORMATION FOR COST CONTROL. Once employees learn that their telephone calls are being monitored, even if it is only a rumor, telephone expense will drop significantly. If nothing happens to substantiate the rumor, however, and reports are not consistently issued on a regular schedule, costs will return in a short time to where they were to begin with.

Once a reporting period has been established, call detail reports should be created and issued as soon as possible after the end of the period. The longer the delay between the time the calls were made and the time they are reported, the harder it will be for callers to remember why they made each call, and the greater the tendency to dismiss the reports as "history".

For the individual telephone user, it is sufficient that his or her manager is reviewing the list of the telephone calls which were made each month to instill a degree of consciousness and frugallity in using the telephone. For the cost center managers and accounting people, however, the computed costs should be tied into actual costs. This appears to create a dilemma, since the bills from the telephone company normally take at least a week or two to arrive and different carriers may bill at different times in the month, and yet to wait until all the bills have arrived before issuing the monthly telephone detail reports conflicts with the need to report the call details promptly to the employees and their managers.

The solution is to use last months telephone bills, which the accounting department has been booking during the past month as they were recieved, and use that total as the total for the month. The difference between the total booked from last months bills, which reflect costs incurred during the prior month, and the current call accounting totals, which reflect the just completed month, can be distributed back across the various costs centers proportionately according to usage. Along with these adjustments-to-actual, other overhead costs, such as labor or equipment amortization, can be included in this distribution.

Thus, each manager is not only charged for actual costs, over which he now has some control, but his share of the overhead will also be proportional to his department's telephone usage.

CONCLUSION. Telephone costs are one of the three or four top expenses in most organizations. Through identifying and accounting for those costs, down to the individual telephone user, and allocating those costs to where they are actually occurring, major cost reductions can be achieved and maintained. These savings can be obtained at reasonable cost with no increase in staff .

Computerized Cargo System

Itzhak Benozer
Israel Ports Authority
74 Derech Petach Tikva
Tel Aviv, Israel

1. BACKGROUND OF ORGANIZATION

The Israel Ports Authority (IPA) controls and operates all the seaports in the State of Israel. Haifa Port (opened in 1933) and Ashdod Port (opened in 1965) are located on the Mediterranean Sea, and Eilat Port (opened in 1965) is located on the Red Sea.

The Ports Authority was established on 1 July 1961 by order of the Ports Authority Law.

The Authority is defined by law as a government corporation whose task is to plan, build, develop, manage, maintain and activate the ports. The three ports are independently managed but are responsible to the general manager; they receive staff services from the main office, which is located in Tel Aviv. They all charge the same rates, which are approved by the IPA Board of Directors (a body consisting of 15 public representatives who represent the users of maritime transport, and by government officials) and by the Finance Committee of the Knesset. They compete between themselves only concerning the standard of services which they give.

The Authority moves about 15 million tons of cargo a year.

The computerized systems of the Ports Authority are based on seven HP-3000 computers located as follows: (1) 3 in Haifa Port; (2) 2 in Ashdod Port; (3) 1 in Eilat Port, and (4) 1 in the main office.

The computers are connected to each other by a DS/3000 via the national communications network "Isranet".

2. FRAMEWORK OF THE SYSTEM

A storage system, or more correctly, a "cargo system" follows up after all cargo moved in the ports, from its unloading from the ship to its being handed over to the customer (import cargo) or from its entrance into the port until its oading onto the ship (export cargo).

The system follows all movement of import and export cargo at the various locations. It covers the following subjects:

1. Input and maintenance of all import manifests.
2. Receipt of cargo to warehouses (loose and containerized).
3. Movement of cargo from location to location within the warehouse.
4. Report on damage.
5. Information center (including customer display terminal).

6. Handling of separated cargo (secondary manifest on international forwarders).
7. Dangerous cargo.
8. Unstuffing of containers.
9. Direct delivery.
10. Input of release documents.
11. Release of cargo (accompanied by gate pass or accompanying documents).
12. Weighing in (all weighing on scales is handled by the system - import, expor, containers).
13. Unclaimed goods.
14. Input of export cargo into the warehouses (storage documents).
15. Input of export documents.
16. Containerizing of cargo.
17. Loading of cargo on the ship.
18. Cancellation of export.
19. Communication with associated systems.

The system is divided up to give the following sub-systems:
1. Manifests (import).
2. Receipt and dispatch (import).
3. Weighing scales (import + export + containers).
4. Unstuffing (import containers).
5. Unclaimed goods.
6. Export.
7. Association with related systems.

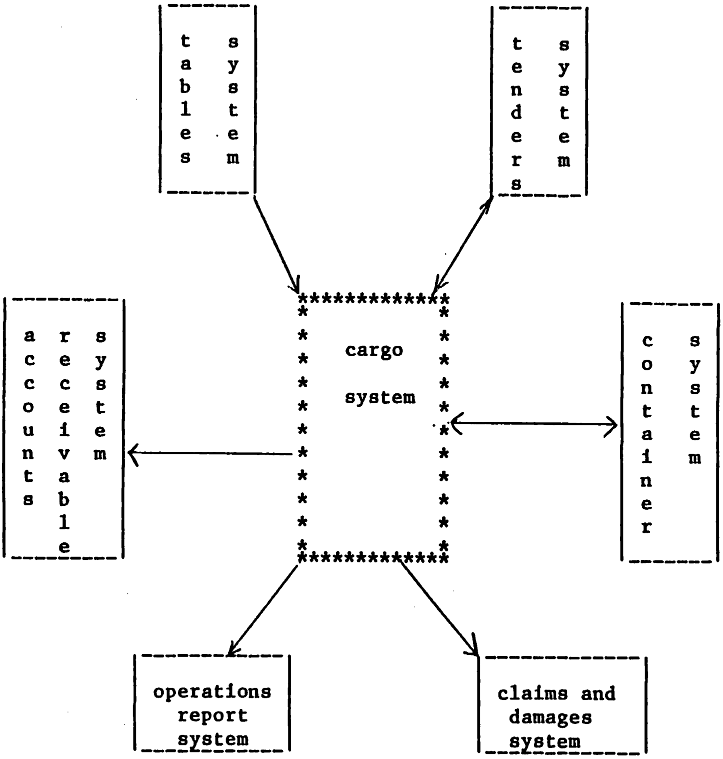
3. CHARACTERISTICS OF THE MANUAL SYSTEM (see table next page)

1. The manual system is characterized by a large quantity of manual work.
2. A large amount of movement of documents and forms between the various locations (transport problems, completion of material, standard of up-dating).
3. The standard of service given to the customer depends on the existing information system - complex procedures, illegible documents, low standard of up-dating.
4. Manual control - difficult to apply.
5. Weak communication with associated system - transfer of forms, loss of control.
6. "Strong" communication with outside factors - a large quantity of paper transferred between outside factors (international forwarders, customs agents, customs, tally companies and shipping agents) and between the Ports Authority (two-way traffic).
7. Disappearance of regulations - increase in deviations.
8. Difficulties in receiving management information.

4. GOALS OF THE SYSTEM

1. Improvement of procedures in cargo system
    a. Elimination of unnecessary activities
    b. Automatic cut-off of accompanying events

Representation of Location of the System and its Communication with Other Systems in the Ports Authority


tables system
tenders system
accounts receivable system
cargo system
container system
operations report system
claims and damages system

   c. Flexibility
   d. Improvement in inspection

2. Improvement of customer services
   a. High up-dating standards
   b. High reliability
   c. High frequency
   d. Reduced waiting time

3. Warehouse management tools

4. Computerized communication to related systems
   a. Containers
   b. Accounts receivable (collections)
   c. Claims and damages
   d. Operations report
   e. Tenders

5. Concentration of all information concerning cargo cancellation procedures in one information center.

5. ADVANTAGES OF THE COMPUTERIZED SYSTEM

a. Savings in Storage Labor

This section is almost totally based on work done by the Management Services and Organization Division of the Main Office, with the people of the Production Engineering Department of the Haifa Port.

- Receipt of magnetic media with manifest details from the Dizengoff Company and its input at the computer unit saves the preparation of documents based on the details of the manifest (packing list).
- There is no need to file packing lists at termination of dispatch (location information on already-dispatched cargo is saved magnetically for a number of years in case clarification is needed).
- Automatic input of data on DOOR containers which have been unpacked saves a range of details needed for the unloading report.
- Automatic production of unloading report.
- Cancellation of location function in the warehouses.
- Decrease of load on location unit due to the customer display terminal at the customs house.
- Receipt of a location terminal based on central information (replaces the procedure of searching through notebooks which need constant up-dating).
- Distribution of list of cargos which have been in the port for more than three months.
- Distribution of turnover reports from the computer eliminates the need to manually fill in various reports in order to calculate premiums.
- Elimination of the need to key-in unstuffing details at the container control center.

- The creation of billable transactions for unstuffing from the system decreases the intervention of the warehouse center in the billing procedure.
- A significant savings in the physical transfer of material from the information center to the warehouses and vice versa.
- There is no need to spend time manually preparing a report of weight deviation and a weight card which are distributed from the computerized system.

These savings were evaluated by means of a survey of some 400 monthly hours and do not include savings which will take place in the future as a result of improvements in the system. For example:

- Connecting the cargo system to the documentation center and to other systems.
- Receipt of manifest details from other agents on magnetic media.
- Connection between the "Zim" computer and the Ports Authority computer and the transfer of data on delivery orders.

b. Improvement of Procedures

- Shortening of procedures because of improvement in up-dating (for example: unstuffing of containerized cargo).
- Decrease of dependence on messengers who transfer material between the various work stations in the Port.
- Possibility of distributing legible and frequent information from the system to the various locations (the accounting department can receive manifest information).
- Exact collection of tariffs (ununitized cargo at unstuffing, dangerous cargo).
- Control of quantities (controlled management of quantities).
- Full follow-up of damages (various damage reports; checking that cargo is not dispatched if only a temporary report has been filled in; saving of historical data).
- Flexibility in dividing cargos between warehosues.
- Flexibility in weighing on the various scales.
- Every change of location will be up-dated and the location system will be immediately informed of the change.
- Faster and more exact charging of unstuffing.
- Full follow-up of cargo on ships operated by more than one agent (manifest number, level of agent and level of ship).
- Follow-up of changes and alterations in the manifest.
- Weight information is transferred automatically at weighing and is therefore more reliable.

c. Improvement of Customer Services

- Fast, exact and reliable receipt of location (immediately following input of receipt information: actual warehouse receiving goods, quantity received and not quantity declared, condition of cargo).
- Customer display terminal permits the customer to receive information about the availability of the cargo.
- Elimination of the location window in the warehouse permits the customer to send his driver directly to the dispatcher at the

warehouse without the need to wait at the warehouse location window.

- The shipping agent receives the reports (unloading report and list of cargo waiting for more than three months) faster. They are more readable and more reliable.
- Shipping agents who supply the Ports Authority with manifest information by magnetic media will receive invoicing information from Accounts Receivable by magnetic media.
- Exact calculation of storage days in the event of two charges.

d. Improved Control

- Excess cargo which was not taken care of will appear on reports which are automatically dispatched.
- The system will check, during dispatch, if the cargo is damaged, and will complete a temporary damage report.
- The quantity on the gate pass will not exceed the available quantity (amount received minus the amount dispatched).
- If the net weight on the dispatch document is greater than that approved on the gate pass, a weight deviation report will automatically be distributed.

6. COMPONENTS OF THE SYSTEM

6.1 Manifest Data Bank

The import procedure begins with the presentation of a manifest which includes details of all the cargo to be unloaded from the ship at the port. Each manifest relates to a ship and each line of the manifest relates to a particular cargo. The cargo can be a loose one (uncontainerized cargo, for example: crate or bulk), it can be part of a container, a Door to Door container. The system also handles transshipment cargo which is unloaded at the port but is designated for another port. The cargo will be reloaded on another ship at a later time.

The manifest is prepared by the shipping agent who collates it from the bills of lading. The data is received by the agent through various means: computer communication, telephone, telegram, telex and facsimile. The agent prepares the material in his office and numbers of the various cargos (a bill of lading cannot identify a cargo as it is issued at the port of loading and is not unique).

The manifest is an official document submitted for customs' approval (this body gives the manifest number). Copies of the signed manifest are transferred to the Ports Authority in order to plan the unloading and storage of the cargo. The manifest should be submitted several hours before the arrival of the ship (in rare cases, it is submitted when work begins on the ship).

Manifest data is fed through a series of screens and is collected in a data base. The data base consists of all the manifest data relevant to the port and a list of operative reports can be produced from it which will be used during unloading (storage document, list of cargo in containers for unstuffing) and a number of management documents.

For each manifest (including ships with no cargo) general details are keyed-in, such as: identification of ship, name of ship, date of arrival of ship, agent identification. Cargo details, including type of cargo, marks and numbers, type of packing, quantity, weight, owner of cargo are listed below the general information. The data is taken from manifests submitted to the storage branch.

Most of the big agents submit the manifests from computerized systems and it is therefore possible to receive the data on magnetic media (this subject will be surveyed in detail further on).

6.2 Direct Delivery

Direct delivery is a procedure in which the cargo is directly transferred from the ship to a vehicle (in unloading) or from a vehicle to the ship (in loading), without being stored. A shipping agent who wishes a certain cargo to be delivered directly must receive approval from the suitable operations level. The warehouse does not handle cargo for which reports are not made in the warehouse. The cargo is registered in the tally report and in the warehouse responsible for delivery and the cargo leaves the port through the gate with accompanying documents prepared by the customs agent.

Some of the cargos go through the weighing scales and are reported from this work station.

6.3 Weighing Scales

The weighing scale is a work station through which some import and export cargos must pass (weighing of cargo is not obligatory and only some of the cargos are weighed at the initiative of customs, the customer or the Ports Authority). The weighing scales also weigh export containers.

The trucks are weighed empty (tare weight) and with cargo (gross weight). The difference between the two weights is the net weight of the cargo.

With each weighing the truck driver receives a document detailing the weight and type of weighing. It should be noted that with import, the truck is weighed for tare before it is weighed for gross and with export the order is opposite.

This work station also handles more complex procedures than the usual ones. For example:

a. One truck with several cargos (must be weighed after loading of each cargo, with the gross weight becoming the tare weight for the next cargo).
b. A truck larger than the scales and weighed in two stages.

c. A truck with one tare weight for several gross weights. The system follows up on the number of weighings.
d. Completion of details when the truck wasn't weighed for tare and it is possible to avoid unloading the cargo and returning to the scales for empty weight.
e. Separating net weight calculated for two cargos which are difficult to separate.
f. Automatic follow-up of cumulative weight that the customer is entitled to release according to his gate pass, and presentation of information about deviations of weight if such exist.

6.4 Receipt of Cargo

All cargo indirectly delivered is stored, at some stage, in a warehouse. At the time of storage, the quantity received and the exact location of the cargo is noted. (It is possible to place the cargo in a large number of locations and every displacement is fed into the system.) If the cargo is found to be damaged, this is noted and details of the damage are given. Damage reports can be given at any stage. If at the time of receipt of the cargo only partial details are given, the system reports that at dispatch, details will be completed and a final damage report will be presented. During the unloading process, a cargo receipt report can be given which details all the cargos received and reported at that moment.

Registration of receipt details based on the data permits receipt of cargo information at the information center. The deviations between expected quantities and received quantities are checked by the warehouse managers and by the representatives of the tally companies (a third party which actually represents the shipping agent and its representatives register all the unloaded cargos, taking into account all the events accompanying the unloading procedure). With the completion of unloading of the ship and clarifications, the system is informed of the date of completion of unloading. This marks the end of unloading and permits presentation of an unloading report which compares the expected quantities with the actually received quantities. The report is made up of several sections, each of which relates to a different group of deviations, for example: missing cargo, excess cargo, uncounted cargo, cargo delivered directly, cargo which fell into the sea, cargo which should be released at a different port. The report is given to the shipping agent who makes corrections on the manifest (the correction is made on a legal document approved by Customs). The corrections are based on the unloading document, the tally report (given to the agent by the tally company), and examinations made by the agent.

6.5 Unstuffing of Containers

Containers slated for unloading (pier) are unloaded at the port or outside of it (at unstuffing terminals). This sub-system only handles containers which are emptied by the port. The unstuffed cargos found in the container are stored in a warehouse in a procedure detailed in

Section 6.4. The unstuffing sub-system transfer container-unstuffing information to the related systems:

a. Transfer of notice to the container system that the container has been unstuffed in order to change the status of the container in the container system.
b. Registration of the fact of unstuffing permits receipt of unstuffing information at the customer display terminal in the information center.
c. Initiation of billing the shipping agent for unstuffing, taking into account the weight of the unstuffed cargo and the type of packing. The billing is input by the collection system which prepares an invoice. The unstuffing sub-system is aided by a packing table which lists next to each type of packing if the cargo was unitized or if there is something which influences the price to be paid.
d. Transactions reports by the unstuffing crew are transferred to the system which calculates premiums.

Communication with related systems which are all computerized is done in real time. This type of communication shortens the procedures, saves the manual transfer of forms, increases the reliability of data, and saves typing time.

6.6 Information Center

Owners of cargo or their forwarders begin the process of cargo release by paying the customs authorities and the port. The customer or his forwarder receives a delivery order from the shipping agent permitting him to release the cargo. Upon completion of the payments to customs and the port, a gate pass number is stamped on a copy of the delivery order, and it is stamped by the customs authorities and the port authorities. These stamps turn the delivery order into a gate pass.

A customer arriving at the information center can check if his cargo is available by using the cutomer display terminal which is at his disposal. This terminal permits the customer to know if his cargo is available or not. The cargo will be identified by the manifest number + order number + secondary number or by the line number of the container in which the cargo is to be found. As the output will only note if the cargo is available or not, the data is not secret, and there is no need for an access restriction.

Upon receipt of a certificate that the cargo is available, the customer goes to the locations information unit where he receives exact information about his cargo (receiving warehouse, specific location and quantity received). At this stage, the details of the gate pass are up-dated to the system.

The system presents a location report which includes all the identifying details of the cargo and full details on the location(s) of the cargo (as noted earlier, the cargo may possibly be in a number of locations). This report permits the driver who comes to collect the cargo to go straight to the dispatching warehouse person without stopping at additional stations on the way.

The information center immediately updates following input of receipt information and unstuffing data, as detailed in the two previous sections.

6.7 Dispatch of Cargo

The driver arrives at the warehouse and takes his gate pass and location report to the dispatcher. The dispatcher checks the cargo against the documents.

If there are no deviations, the cargo is loaded on the vehicle. The dispatcher updates the system with details of the dispatch (the dispatched quantity is erased from the location from which it was taken, the dispatched quantity is updated, and details of the vehicle and the driver are registered). The driver goes to the port gate and leaves the gate pass there.

When cargo is only partially removed (when the entire cargo cannot be loaded onto one vehicle) the driver leaves the gate pass at the gate on his first trip and all the other trips are accompanied by an accompanying pass prepared by the computerized system which follows all quantities which are dispatched. The accompanying pass is attached to the gate pass at the gate. The documents accumulated at the gate are periodically transferred to the collection department to check if additional invoicing is necessary (especially because of additional storage time).

6.8 Unclaimed Cargo

Cargo which has not been released three months after its receipt can be sold. The system will periodically release a list of cargo which has been in the port for more than three months. The list will include all details identifying the cargo and its exact location. Every warehouse manager will receive a list of the cargos found in his warehouse. He will check his warehouse and confirm that these cargos are physically in their place.

In the event that the cargo is not located, documents in the port and in the offices of the shipping agent will be examined. If these checks show that the cargo has been dispatched, this will be sent to the system. If no confirmation of dispatch is found, a theft report will be filed (one type of damage report). These two procedures will prevent this cargo from appearing on the existing inventory report of the warehouses. The cargos will be checked to see if they can be given the status of unclaimed cargo. Cargos that can be labelled as unclaimed, will be transferred to the unclaimed cargo warehouse. With the transfer of the handling to the unclaimed cargo warehouse, the owner of the cargo and the shipping agent will be notified by letter that the cargo stands to be sold.

If no reply is received from the customer within 14 days, the various authorities (customs, Ministry of Health, Ministry of Agriculture, Ministry of Transport) are asked if the cargo can be sold as unclaimed cargo. Upon receipt of approval to sell by tender, the cargo is opened and listed down to its smallest detail. The description is transferred to the tenders system, which presents a list of cargos which can be sold. The tenders system handles the advertising of the tender, the collecting of bids and the winner of the tender, and transfers the results of the tender to the cargo system, so that it will release the goods to the winner of the tender.

6.9 Direct and Indirect Export

In this procedure the cargo arrives by vehicle to the port's export warehouses. The cargo is accompanied by cargo documents prepared by the customs agent. The warehouse people check the cargo and input its identifying details and its location through the terminal in the warehouse. Upon completion of input, the system will print a storage document which will be considered a certificate of receipt to the warehouse.

When the time comes for loading to begin, the customer or his forwarder presents an export document, which gives him permission to remove the cargo from the warehouse for loading. A good part of the cargo is transferred to the area where containerization is done (loading of cargo into export container). Upon completion of stuffing of the cargo into the container, the container is transferred to the container terminal (the system accepts the identification of the container into which the cargo is stuffed).

If the cargo is exported by means other than container, it is transferred to the loading dock. This transfer will be reported to the system.

The system communicates between storage documents and export documents. The cargos loaded onto the ship are registered by a tally company employee. The registrations are checked against the documents at the port and if they are suitable, a cargo listing can be prepared.

Direct export is analogous to direct import, when the cargo arrives by truck directly to the loading dock without being warehoused. The only accompanying document in this case is the export document.

7. CHARACTERISTICS WHICH UNIFY THE SYSTEM

7.1 Communication with Outside Factors

This system, which is the largest operations system existing in the Ports Authority today, is characterized by its strong communication with outside factors. This communication is expressed by intensive daily contact between representatives of these factors and cargo workers at the port, and by massive transfers of documents in both directions.

a. Shipping agent
   Submission of manifest
   Submission of notification of dangerous materials
   Request for direct delivery
   Receipt of unloading report
   Submission of alterations to the manifest
   Obtaining of delivery order
   Receipt of list of cargos not released within three months.

b. Customs agent
   This factor represents the customer in the procedure of releasing the cargo, from the moment the customer pays the shipping agent (it should be remembered that some customers do not use the services of a customs agent and handle release of goods by themselves).
   His activities involving the system are:
   Receipt of delivery order from the shipping agent and its transfer at the port.
   Receipt of information about the cargo and receipt of a location report at the information center.
   Handing of the location report to the driver, who drives to the warehouse.
   Issuing accompanying documents for bulk cargo for direct delivery.
   Handling weight deviations beyond those listed in the gate pass.

   Preparing of export documents.

c. Customs
   Receipt of copy of manifest with printed number and signature.
   Certification of alterations on manifest.
   Demand for customs examination.
   Cancellation of stoppage of cargo.
   Signature on gate pass.
   Certification of sale of unclaimed cargo.
   Determination of value for customs purposes, which will influence the port tariff.

d. Forwarders
   The international forwarder brings cargo from a number of customers, on one bill of lading. With the arrival of the container at the port, he takes care of paying the shipping agent and receives from the agent a delivery order which he turns in at the port. The forwarder attaches a divided list detailing the entire cargo including the general delivery order, and he supplies his individual customers with secondary delivery orders which relate to the cargos listed in the divided list.

e. Tally companies
   Representatives of these companies station themselves near the pier and register all cargo unloaded from the ship or loaded onto the ship. At the stuffing and unstuffing terminal, they register the cargo removed from the containers or stuffed into them.
   The tally company performs a number of activities connected with the system, such as registration on-the-spot of quantities on forms

provided by the system, such as: storage report or packing list; clarification in the event of unsuitability of data, submission of centralized tally report.

7.2 Transfer of Data To and From Outside Factors on Magnetic Media

a. "Zim", the national shipping company, handles 50% of the cargo moved in the ports of Israel. In light of this fact, the IPA made a trial attempt to receive data from the shipping agent on magnetic tape. Starting in January 1984, a magnetic tape is transferred daily with the manifest data obtained by the data processing unit of the Dizengoff Company (which represents Zim) from Zim by the data processing unit of the port of Haifa. The tape is called by software which transfers the data to the data base in which other manifest data is stored by means of Data Entry.

With the beginning of the operation of a cargo system at Ashdod Port in February 1988, magnetic tapes were transferred with manifest data on ships coming to Ashdod Port. The tapes are transferred to the computer unit at Haifa Port, called to the Haifa computer, and then transferred to the Ashdod Port computer by means of the National X-25 network via DS-X25 communications software.

b. The Dizengoff Company presents a magnetic tape every day with data concerning expected containers at the Ashdod Port container terminal. This tape is transferred to the computer unit at Haifa Port and is handled in a similar way to the manifest tape, as described above.

c. Recently a new trial has been started to input a diskette created on a personal computer by an additional shipping agent. The diskette should include manifest data relating to cargos which should arrive at Haifa and Ashdod ports. The diskette is read by software which operates on a PC located in the computer unit in Tel Aviv and the information is then uploaded to the HP-3000.

d. The Dizengoff Company receives a tape once a week with all the data of billing done by the Ports Authority to the Dizengoff Company. The various systems handling collection from shipping agents obtain these tapes, which include data relating to invoices of this specific agent. The tapes are presented at each port to the local representative of the agent, who transfers them to the correct unit at Zim in Haifa, for handling. The transfer of data by magnetic media reduced an enormous amount of keying-in data and decreased the backlog in preparing financial reports of the Dizengoff Company.

7.3 Connection between Computers

In parallel to the transfer of tapes between the Zim Company and the Ports Authority, it was decided to check the possibility of connecting the Zim Company computer to the Ports Authority computer in Haifa

Port. The purposes of this connection are as follows:

a. Independence from messengers who physically had to transfer the tapes.
b. Transfer of data at high frequency (once every 20 minutes). This will permit input of data on delivery orders in a reasonably short time from the moment of their creation. The creation of a temporary pool of delivery order data arriving through communication will improve service to the agent's customers (it will save keying-in time of data at the port and will prevent errors because of incorrect input). Connection between the computers should fulfill these two purposes.

The problems with the connections are in two areas:

a. Different hardware in the two companies (HP-3000 at the Ports Authority and VAX-8250 at Zim) to be solved through the use of a PC as an intermediary.
b. Approach to information and data should be protected on the Ports Authority computer, as the Ports Authority is a public company subject to the Privacy Protection Law.

7.4 Information Center

The system is charcterized by the strong daily communication with a large number of customers. In the past, customers would receive operational reports based on handwritten reports which were partial and inexact (based on data of expectation and not on actual data). Obtaining the data was done manually.

The establishment of a computerized system based on work stations at every operational location made it possible to turn the locations unit into an information center based on data existing in computerized systems. A customer receives the location of a Door to Door container from the container system and the location of a containerized cargo or a loose cargo from the cargo system. The menu on the screen includes several lines which suit the types of dispatch and any activity chosen will connect that terminal to the suitable system (the connection between two different systems on the same computer is made transparent for the user).

The information center was recently expanded by the addition of a customer display terminal, which will help customers make clarifications concerning their cargos. This terminal, which is located in a room specifically for customers, will be operated by the customers themselves and will permit them to know if their cargos are available or not. The limitations of information given about the cargo do not necessitate checking if the customer is eligible to receive the information and it is suitable to the management of the Privacy Law of availability of information. The customer can request information according to container in which his cargo will be found or according to identification of the cargo (manifest + number + secondary).

This service can be extended to othr areas and terminals can be located at the offices of large customers so that they can thus serve their customers in their offices without making it necessary for their customers to come to the port.

7.5 Intensive Communication with Other Computerized Systems at the Ports Authority

The cargo system is characterized by its strong communication with other computerized systems in operation today.

a. Container system

Receipt of data on Door and Pier containers which were unloaded (a program which surveys the container file and outputs, according to manifest, all the Door and Pier containers unloaded from a particular ship. Door type containers will have their receipt noted so that an unloading report can be prepared, and Pier type containers will be registered in the file of containers for expected unstuffing).

Locations screens connected to the container system.

Transfer of information about unstuffing of containers from the cargo system to the container system.

Preparation of an expected container file within the manifest.

b. Collection system

Transfer of unstuffing data.

Transfer of labelling data to the cargo servicing accounting system.

Transfer of moving and storage data to the cargo servicing accounting system.

Transfer of notice of dangerous materials.

Exact collection based on actually received quantities.

c. Tenders system

Transfer of description of cargo to be sold by tender.

Input of results of tender into the cargo system.

d. Operations report system

Transfer of turnover data of workers.

e. Claims and damages system

Transfer of data on damaged cargo.

f. Tables system

The cargo system uses about 20 tables which are found in the central tables system.

7.6 Communication between the Computers

a. Communication between different computers located at the same location.

This connection, based on HP LAN solution (IEEE 802-3), permits the existence of communication between the cargo system and the other computerized systems which are in operation on other computers.

b. Communication between computers in various ports.

This type of communication is performed via the national X-25 communictions network, "Isranet", based on DS-X-25 and will be replaced by the end of the year by NS-X-25. This permits the following activities:

1. Transfer of manifest and expected container data from the Haifa port computer to the Ashdod port computer.
2. Transfer of manifest data which were slated for one port to another port following a last minute change in the port of arrival.
3. Comparison of missing cargo at one port with excess cargo at the other port.
4. Transfer of data on cargo accepted at one port, but released at the other port (the cargo will be transferred before the release procedure from port to port, overland).

7.7 Computerized Work Station at the Weighing Scales

Every port operates a number of weighing stations (used to weigh bulk or other cargo whose weighing is demanded by the customs authorities or by the owner of the cargo).

The weighing scales are equipped with a computerized weighing head with an RS-232-C communications interface. The weighing head transfers the weight data to the HP-3000 computer at a rate of 2400 BAUD, upon request of the HP-3000. Weight data is input by means of two screens (one for weighing before the cargo is handled by the cargo system and the second after handling and filling in the suitable documents). Each one of them has a field into which the weight is transferred in real time by Authority written communications software (in the event of a communciations breakdown, the data will be fed manually). The work station at the weighing scales performs the following operations:

1. Tare weight of the truck.
2. Gross weight of the truck.
3. Distribution of weighing documents.
4. Distribution of notice on weight deviation.
5. Distribution of weight chart to gate pass.
6. Query on tare weight.
7. Distribution of report on turnover of the weighers.

8. EQUIPMENT AVAILABLE

Haifa Port

	Existing & Operating		Additions in Progress	
	Terminal	Printer	Terminal	Printer
Information Center	3	2		
Warehouse 7	1	1		
Warehouse 15	1	1		
Unstuffing Terminal	3	1	3	1
Scales, East	1	1		
Locations	1	1		
Customs House			1	
Documentation Center			1	1
Kishon 3	1	1		
Scales, West	1	1		
Scales,Kishon			1	1
Warehouse 3			1	1
Total	12	9	7	4

Ashdod Port

Information Center	3	1		
Locations			3	1
Documentation			1	1
Warehouse Manager			1	1
Weighing Scales			3	3
Warehouse 201			1	1
Warehouse 104			1	1
Warehouse 105			1	1
Warehouse 202			1	1
Warehouse 203			1	1
Warehouse 207			1	1
Warehouse 302			1	1
Warehouse 303			1	1
Total	3	1	16	14

9. SOFTWARE

9.1 Overview

Sub-system	Application at Haifa Port	Application at Ashdod Port
Manifests	January 1984	January 1988
Receipt and Despatch	June 1986	June 1988
Weighing Scales	January 1986	June 1988
Unstuffing	June 1988	December 1988
Unclaimed Goods	August 1988	January 1989
Export	January 1989	June 1989
Communication with Related Systems	January 1989	June 1989

9.2 The System Files

The cargo system is based on the Turbo Image/V data base. The data base includes:

10 manual master files
5 automatic master files
15 detail files
Average number of keys for Detail file is 2.
Maximum number of keys for Detail file is 3.
The maximum number of pointers from a Master file is 5.
The capacity of the Data Base is 250 megabytes and it is capable of reaching 400 metabytes.

Other than the Data Base files, there are about 10 KSAM files and 10 MPE files. One of the KSAM files is the central tables file, which today includes about 80 tables, of which about 20 serve the cargo system.

9.3 System Programs

Development is mainly based on Powerhouse software (QUICK, QUIZ, QTP) version 5.01. So far, some 60 QUICK programs, 60 QUIZ programs and 10 QTP programs have been developed. The QUICK programs, which handle complex screens, are very large and consist of 1700 lines of code. Most of the programs have 300-500 lines of code.

In addition to Powerhouse software, programs written in third generation languages are used, such as: COBOL and SPL. Programs written in COBOL are BATCH programs which should perform transfers and/or massive processing of data.

One program was written in SPL, and its function is to transfer the weighing data from the weighing head to the field located on the screen where weighing data is input. The QUICK program performs dispatch reading which is located in a segmented library. The reading is done by the instruction Do External. The weighing data is input as ASCII String and transferred to QUICK as Integer.

9.4 Personal Computer Software

The system is assisted by PC software services. This software should perform the following functions:

1. Transfer of manifest data written on diskette to the HP-3000.
2. Connection of HP-3000 to the VAX by means of a PC. For this purpose, a software package (Reflection*, from Walker, Richer and Quinn, Inc.) is used which performs emulation on the PC, at a terminal recognized by the HP-3000 (HP-2622) and as a terminal recognized by the VAX (VT-100), plus file transfers to and from each computer.

9.5 Summary

1. This application is very complex and has a large number of exceptions which were not evident at the beginning. The decision to use the Powerhouse software as a development tool for the system was correct. The system developed as a prototype. Use of a third-generation language would significantly increase the reaction time to any changes required in the applications process.
2. In a number of cases it was decided to substitute COBOL for the QUIZ and QTP programs because the functions were simple to program and easy to maintain.

HYPOTHESIS DRIVEN PROGRAMMING

Ross G. Hopmans
Brant Computer Services Limited
2605 Skymark Avenue
Mississauga, Ontario
CANADA L4W 4L5
(416) 238-9790

INTRODUCTION

In this paper I will introduce the basics of Artificial Intelligence, Knowledge-Based systems, Logic Programming and the Prolog language to establish the groundwork for, and a means to implement goal-oriented or Hypothesis Driven programming. Through the use of examples I hope to show the mechanics and benefits of this technology.

ARTIFICIAL INTELLIGENCE

The term ARTIFICIAL INTELLIGENCE is generally used to describe the technology for dealing with KNOWLEDGE (a superset of DATA).

Artificial Intelligence is concerned with making machines more useful by causing them to mimic processes that, if performed by humans, would be considered intelligent. Many countries have significant, national-level Advanced Information Technology programs underway. This approach, called "technology push", focuses on the development of the technology for yet unspecified commercial applications. In contrast, most of the successfully implemented AI applications have resulted from demand or "market pull" for new solutions to existing problems.

AI is concerned with the study of systems that represent, acquire and use knowledge in order to perceive, reason, plan, act and use language. Some differences between biological and artificial intelligence are as follows:

Biological Intelligence	*Artificial Intelligence*
perishable	*permanent*
difficult to transfer	*easy to duplicate*
erratic	*consistent*
difficult to reproduce	*easy to document*
creative	*uninspired*
learned	*programmed*
sensory input	*symbolic input*
wide context	*narrow focus*

KNOWLEDGE-BASED AND EXPERT SYSTEMS

Concentrating on the simulation of human expertise has produced the most successful AI programs in the form of expert systems that help make decisions by exploiting scarce or expensive expertise. They require a lot of high quality, specific knowledge about a particular topic.

Some characteristic categories are:

Interpretation	*inferring situation descriptions from sensor data*
Prediction	*inferring likely consequences of given situations*
Diagnosis	*inferring system malfunction from observations*
Monitoring	*comparing observations to expected outcomes*
Advisor	*expressing opinions after drawing from relevant facts*

An expert system can make deductions through inference, motivate its own actions and solutions, suggest alternatives to solutions and handle uncertain information.

The terms "Knowledge-based systems" and "expert systems" are often used interchangeably, but their meanings are quite different. The term knowledge-based systems describes an important technical issue: it indicates that the source of the program's power is primarily a large body of task-specific knowledge. Such systems may function in a variety of roles acting as assistants, colleagues and sometimes as experts.

The term expert systems refers primarily to an aspiration - the desire to have a system that works as well as a human expert.

Most of the interesting things we know about the world are not numeric and most of the knowledge we have about the world is not well modelled with arithmetic. We think and reason about problems using IF/THEN rules, but how do we capture these type of rules using arithmetic?

Knowledge-Based systems take a particular view of the answer to the question "Why are experts experts?" Do they think faster than the rest of us? Do they think differently? Do they have a general purpose trick of thinking or problem solving?

To call something a knowledge-based system subscribes to the belief that experts are experts because of what they know. Expertise arises from knowledge. The power of knowledge-based systems comes from their base of knowledge about the specific task at hand.

A Knowledge-Based Expert System is a close approximation to cloning. What we can not do biologically, we try to do intellectually. We take take rare and valuable expertise and try to clone it by talking at length with the person who has it. It is useful because almost all organizations have a knowledge bottleneck - a place where its productivity is limited by a scarcity of knowledge and skill.

Who do you miss when they are sick? Who would you hate to lose to the competition? Whose impending retirement has you worried? Who do you wish you had five more of?

Cloning that knowledge is one possible remedy to these problems and knowledge-based systems offer us the possibility of doing that. The focus of knowledge-based systems is on *knowledge* - collecting it, formalizing it and putting it to work in the computer.

WHY USE KNOWLEDGE-BASED SYSTEMS?

The oldest expert system in commercial use configures computer systems. But that type of problem is not unique to the electronics or manufacturing industries. Contracts built from standard clauses and insurance policies built from standard coverage also need to be "assembled". In all cases we want to ensure that all the needed parts are there, nothing is omitted, nothing extra is added and they all function together smoothly. So this technology can span across industry sectors.

There is also an issue of corporate memory. We would like to retain human skill and have it outlive any single individual. Using knowledge-based technology we can debrief the experts who have a lifetime of experience and capture a useful part of what they know - making more real the idea of corporate memory.

If the line of responsibility in a knowledge-based system falls mainly to the human, then the program is more of an assistant. If we can push that line of responsibility more to the machine end and make the program smarter then that may be something we can call an expert system. Useful applications can fall anywhere along that line. We can take an evolutionary approach and start a program out as an assistant and add more knowledge over time to give the machine more responsibility.

HOW DO KNOWLEDGE-BASED SYSTEMS WORK?

Knowledge-based systems are characterized by their reliance on large stores of rule-based knowledge as a basis of their expertise. In effect, we have done a dicing of the human expert's knowledge into individual rules. Rules are comprehensible and each stands on its own feet. Each rule makes one small decision that relays one thing to do under a particular set of circumstances. We use a rule whenever it is relevant - that is whenever its pre-conditions (the items in the IF part of the rule) have been met.

In traditional programming we write complete, decision-making procedures, whereas in AI we write individual decision rules. Traditional programming tells the computer what to do; here we tell it what to know. It turns out that a few hundred to a few thousand rules are adequate for capturing interesting, valuable and non-trivial skills.

A big difference between an expert system and a conventional system is that knowledge is stored in a knowledge base in the expert system but coded into a program in the conventional approach. Program code is a very primitive way of storing knowledge. It is rigid, static and inaccessible.

The art of building knowledge systems is one of collecting and accumulating facts and rules. This collection comprises the knowledge base. The other component is the inference engine - that part of the program that uses the rules of the knowledge-based system and applies them to the problem at hand. The engine's primary job is to do symbolic inference. The separation of knowledge base and inference engine is characteristic of knowledge systems and but is atypical of traditional systems.

Knowledge-based systems can generate an explanation of their behaviour and the answer is comprehensible because the computer is doing symbolic inference and not arithmetic; reasoning and not calculation. We have turned the computer into a logical reasoning engine - a device whose basic operation is rule retrieval and application. Traditional applications are not, nor were they intended to be models of human problem solving. Instead, they are powerful and have an important place but they are not necessarily comprehensible.

As a result, knowledge-based system have a property we call transparency. When you look inside them, their operation makes sense to us. They are not black boxes. They reveal their line of reasoning and it is sensible.

Eventually a knowledge-based system reaches a conclusion with a list of plausible alternatives for decision support. It does not just pick out one answer it thinks is best. Transparency allows us to ask how the program decided what is plausible and it allows us to examine the reasoning chain because it keeps an audit trail of the logic it uses and allows us to examine that audit trail in as much detail as we like.

LOGIC PROGRAMMING

The study of symbolic logic goes back to the work of Aristotle in the fourth century B.C. First order predicate logic is a branch of symbolic logic that has evolved largely in the twentieth century. It is a universal, abstract language for representing knowledge and solving problems. Logic programming is based on a subset of first order predicate logic.

Aristotle attempted to codify into a scientific system the way that knowledge could be most effectively pursued through rational debate. He wanted to establish a standard whereby the correctness of a line of reasoning could be established.

For instance, given the following two premises:

> *Socrates is Human.*
> *Humans are Mortal.*

then the following conclusion is valid:

> *Socrates is Mortal.*

The process of reaching a conclusion from the premises is also called making an inference. The above inference takes place because the predicate class of one statement matches the subject class of another. As a result of the inference, a new statement is formed out of the subject of one statement and the predicate of the other.

Logic provides a precise language for the expression of one's goals (or hypotheses), knowledge and assumptions. Logic provides the foundation for deducing consequences from premises, for studying the truth or falsity of statements given the truth or falsity of other statements, for establishing the consistency of one's claims and for verifying the validity of one's arguments.

Through Logic Programming we can *prove* program correctness!

The above example can be proven correct through the simplest and most powerful of Aristotle's inference rules known as *modus ponens* which states:

> *If P implies Q and P is true, then Q is true.*

Restating our example in this format, we have a rule P implies Q, and a fact P as:

> *If a person is human then that person is mortal.*
> *Socrates is human.*

or, more programatically familiar:

> *If human(X) then mortal(X).*
> *human(Socrates).*

We can conclude that *Socrates is Mortal.*

In this example we applied a fact to a simple, comprehensible, general-purpose rule and we were able to infer a new piece of information which we can prove to be correct.

Although computers were intended for use by humans, the language for expressing problems to the computer and instructing it how to solve them was designed from the perspective of the engineering of the computer alone.

Logic programming departs radically from the mainstream of computer languages. Rather than being derived from the von Neumann machine model, it is derived from an abstract model, which has no direct relationship or dependency to one machine model or another. It is based on the belief that instead of the human learning to think in terms of the operations of a computer, which some scientists and engineers at some point in history happened to find easy and cost-efficient to build, the computer should perform instructions that are easy for humans to provide. Logic programming suggests that explicit instructions for operation not be given but rather the knowledge about the problem and assumptions that are sufficient to solve it be stated explicitly. This constitutes an alternative to the conventional program. The logic program can be executed by providing it with a hypothesis (or problem), formalized as a logical statement to be proven, called a goal statement. The execution is an attempt to solve the problem - that is, to prove the hypothesis given the assumptions in the logic program.

A major aim of logic programming is to enable the programmer to program at a higher level. Ideally one should write axioms that define the desired relationships, maintaining ignorance of the way they are going to be used by the execution mechanism. Current logic programming languages such as Prolog, however cannot ignore how their execution mechanisms work. Effective logic programming requires a certain implementation knowledge for effective execution.

PROLOG

Fifth generation languages are valuable because they help reduce the distance between the verbal description of a process and its representation in executable code. The resulting programs are thus easier to both read and change. The logical foundations of a program tend to be closer to the surface in fifth generation languages than in more conventional ones.

The evolution of computer languages is an evolution away from low-level languages, in which the programmer specifies how something is to be done, toward high-level languages, in which the programmer specifies simply what is to be done. Most languages, Lisp included, are "how-to" languages. Prolog breaks away from that, encouraging the programmer to describe situations and problems, not the detailed means by which the problems are to be solved.

Prolog is a programming language centered around a small set of basic mechanisms, including pattern matching, tree-based data structuring, and automatic backtracking. This small set constitutes a surprisingly powerful and flexible programming framework.

A Prolog program consists of a set of rules for deducing the truth of a given hypothesis (or goal) from a conjunction of other hypotheses. A set of rules with the same head represents alternative ways of establishing the same hypothesis. Such a set is sometimes called a procedure. Prolog procedures are somewhat analogous to procedures in mainstream languages and are conceptually like the *Case* statement. The goals in the body of a procedure are invoked sequentially, and a procedure is exited when all hypotheses in one of its rules have succeeded.

However, a fundamental difference in the flow of control arises when one of the hypotheses fails. Whereas in mainstream languages failed statements must be handled explicitly, Prolog automatically backs up to the previous hypothesis, attempting to satisfy it in a different way. This behaviour is known as backtracking.

Programming in Prolog often consists of merely describing essential properties of a problem, defining the relations and rules applicable to the problem and stating known facts relevant to the problem. With conventional programming languages, the programmer has to spell out a detailed sequence of steps which the computer must perform. The declarative style of programming made possible by Prolog is faster, easier and less error-prone.

Prolog's symbolic nature, dynamic memory management and flexible structure makes it an ideal language for the rapid prototyping of virtually any kind of system but Prolog is especially well suited for problems that involve objects - in particular, structured objects - and relations between them.

The control structure in Prolog is unification, or a pattern matching process operating on a sophisticated internal data base which contains a full relational database. Data and programs are stored in this internal database which in searched for solutions to goals. The Prolog system can search for alternative solutions (called non-determinism) by backtracking through the search space. Prolog programs in the database consist of statements which express relationships between entities. The "logical engine" in the Prolog system then runs the program by inferring true statements from the given relationships.

Unlike conventional languages, Prolog incorporates program control into the language itself. The programmer is relieved of the majority of the control of program flow and can focus on expressing data objects' relationships.

Prolog is not "just another language"; it may by THE language of the future. Prolog is based on logic, and not on a mapping of the machine architecture. Prolog provides a *Procedural Interpretation* which defines how the problem should be solved with an algorithm and a *Declarative Interpretation* which describes the problems and what is known. It is the Declarative Interpretation allows us to prove programs correct.

There is currently no universal Prolog standard. Over time, the implementation described by W.F. Clocksin and C.S. Mellish in *Programming in Prolog* (Springer-Verlag, 1985) has emerged as the de facto standard.

HOW PROLOG WORKS

The control in Prolog programs is like in conventional procedural languages as long as the computation progresses forward. Hypothesis invocation corresponds to procedure invocation, and the ordering of hypotheses in the body of clauses corresponds to sequencing of statements. The differences show when backtracking occurs. In a conventional language, if a computation cannot proceed (eg. all branches of a case statement are false) a runtime error occurs. In Prolog, the computation is simply undone to the last choice made, and a different computation path is attempted.

The data structures manipulated by logic programs (terms) correspond to general record structures in conventional programming languages. The handling of data strucures is very flexible in Prolog. Like LISP, Prolog is a declaration free, typeless language. Logical variables refer to individuals rather than memory locations. Consequently, having specified a particular individual, the variable cannot be made to refer to another individual. In other words, logic programming does not support destructive assignment where the contents of an initialized variable can change.

A question to Prolog is always a sequence of one or more hypotheses. To answer a question, Prolog tries to satisfy all the hypotheses. To satisfy a hypothesis means to demonstrate that the hypothesis is true, assuming that the relations in the program are true. In other words, to satisfy a hypothesis means to demonstrate that the hypothesis logically follows from the facts and rules in the program. If the question contains variables, Prolog also has to find what are the particular objects (in place of the variables) for which the hypotheses are satisfied. If Prolog cannot demonstrate for some instantiation of variables that the hypotheses logically follow from the program, then Prolog's answer to the question will be 'no'.

Prolog accepts facts and rules as a set of axioms and the user's question as a conjectured theorem; then it tries to prove this theorem - that is to demonstrate that it can be logically derived from the axioms.

The graphical illustration of an execution trace has the form of a tree. The nodes correspond to hypotheses and the arcs to the application of alternative program clauses that transform the hypotheses at one node into hypotheses at another node. The top hypothesis is satisfied when a path is found from the root node to a leaf node labelled "yes". A leaf is labelled "yes" if it is a simple fact. The execution of Prolog programs is the searching for such paths. During the search Prolog may enter an unsuccessful branch. When Prolog discovers that a branch fails it automatically backtracks to the previous node and tries to apply an alternative clause at that node.

DECLARATIVE VS. PROCEDURAL

We distinguish between two levels of meaning of Prolog programs; namely the declarative meaning and the procedural meaning.

The procedural meaning is concerned only with the relations defined by the program. The declarative meaning thus determines what will be the output of the program. On the other hand, the procedural meaning also determines how this output is obtained; that is, how are the relations actually evaluated by the Prolog system.

Consider the clause P :- Q,R. Some alternative declarative readings of this clause are:

> *P is true is Q and R are true.*
> *From Q and R follows P.*

Two alternative procedural readings of this clause are:

> *To solve problem P, first solve subproblem Q, then subproblem R*
> *To satisfy R, first satisfy Q and then R*

The procedural reading do not only define the logical relations between the head of the clause and the hypotheses in the body, but also the order in which the hypotheses are to be processed.

The declarative meaning of programs determines whether a given hypothesis is true, and if so, for what values of variables is it true.

The procedural meaning specifies how Prolog answers questions.

P :- P is declaratively quite correct but procedurally useless and can cause problems to Prolog as it results in a infinite loop. What is unusual about Prolog is that the declarative meaning of a program may be correct, but the program is at the same time procedurally incorrect in that it is unable to produce an answer to a question.

A general practical heuristic in problem solving is that is it often useful to try the simplest idea first.

The reason we should not forget about the declarative meaning is that progress in programming technology is achieved by moving away from procedural details toward declarative aspects, which are normally easier to formulate and understand. The system itself, not the programmer, should carry the burden of filling in the procedural details.

PROGRAMMING IN PROLOG

Prolog is related to mathematical logic, so its syntax and meaning can be specified most concisely with references to logic. But these concepts are not necessary for understanding and using Prolog as a programming tool.

Often the key step toward a solution is to generalize the problem. By considering a more general problem, the solution may become easier to formulate.

Prolog programs consist almost exclusively of declarations. The programmer is able to leave many of the control decisions to the problem solving mechanism. SQL has a similar declarative nature in that the user specifies what type of answer is required and the interpreter decides how to supply it.

Consider the following example Prolog program to to describe why a product benefits a customer. The program consists of 14 statements: 3 rules and 11 facts.

```
PRODUCT benefits CUSTOMER if
        CUSTOMER is-involved-with BUSINESS and
        PRODUCT improves BUSINESS.
PRODUCT benefits CUSTOMER is
        CUSTOMER employs PROFESSIONAL and
        PRODUCT increases-productivity-of PROFESSIONAL.

PRODUCT improves BUSINESS if
        BUSINESS requires TOOL and
        PRODUCT enhances TOOL.

hewlett-packard is-involved-with computers.
westinghouse is-involved-with power-plants.
metropolitan-life is-involved-with insurance-policies.

power-plants requires regulatory-analysis.
insurance-policies requires insurance-underwriting.

prolog enhances expert-systesms.
prolog enhances natural-language-processing.
prolog enhances insurance-underwriting.
prolog enhances regulatory-analysis.

hewlett-packard employes software-engineers.

prolog increases-productivity-of software-engineers.
```

Most Prolog implementations support the "syntactic sugar" used here to enhance the readability of the program. The equivalent, standard Prolog code is as follows:

```
benefits(PRODUCT,CUSTOMER) if
        is-involved-with(CUSTOMER,BUSINESS) and
        improves(PRODUCT,BUSINESS).
benefits(PRODUCT,CUSTOMER) if
        employs(CUSTOMER,PROFESSIONAL) and
        increases-productivity-of(PRODUCT,PROFESSIONAL).

improves(PRODUCT,BUSINESS) if
        requires(BUSINESS,TOOL) and
        enhances(PRODUCT,TOOL).

is-involved-with(hewlett-packard,computers).
is-involved-with(westinghouse,power-plants).
is-involved-with(metropolitan-life,insurance-policies).

requires(power-plants,regulatory-analysis).
requires(insurance-policies,insurance-underwriting).

enhances(prolog,expert-systems).
enhances(prolog,natural-language-processing).
enhances(prolog,insurance-underwriting).
enhances(prolog,regulatory-analysis).

employs(hewlett-packard,software-engineers).

increases-productivity-of(prolog,software-engineers).
```

The words in upper case are Prolog variables. Variables are local to a statement so that each rule can be understood in isolation.

In this program, "benefits" and "improves" are rules whereas "is-involved-with", "requires", "enhances", "employs" and "increases-productivity-of" are all facts. Prolog, however, does not differentiate between facts and rules. In other words, we could quite correctly add a new fact:

benefits(prolog,brant).

This Prolog program contains a knowledge base of facts and rules which we can use in many different ways. To execute any portion of the program, we give it a hypothesis or goal. The following hypotheses succeed simply because they exist in the fact base:

enhances(prolog,expert-systems).
is-involved-with(hewlett-packard,computers).
employs(hewlett-packard,software-engineers).

Similarly, the following hypotheses fail because they cannot be proven correct:

enhances(prolog,data-processing).
employees(brant,software-engineers).

Using the rules, we can hypothesize about a particular solution, such as:

(1) benefits(prolog,hewlett-packard)

or our goal may be to find a solution, such as:

(2) benefits(prolog,CUSTOMER).

In the case (1), the hypothesis would succeed because the second "benefits" rule can be successfully applied to the facts provided. In case (2), the hypothesis would succeed, returning the first value of the variable CUSTOMER for which it did succeed. In this case, the value returned would be "westinghouse".

Through its backtracking capability, Prolog is able to return all the values of variables for which the hypothesis succeeds. For example,

benefits(prolog,CUSTOMER), fail.

would locate all instances of CUSTOMER for which the hypothesis is true. By explicitly telling the hypothesis to fail after successfully finding a value for CUSTOMER, the inference engine will attempt to solve for the hypothesis in all possible ways.

In fact, using:

benefits(PRODUCT,CUSTOMER), fail.

Prolog will find all PRODUCT/CUSTOMER touples for which the hypothesis is true.

This small, simple example program begins to illustrate the power of this type of programming. The programmer tells Prolog what he knows. He provides the program with facts and rules and he can then hypothesize about anything in the knowledge base.

It is the inference engine and not the programmer that takes on the burden of solving the problem by doing the symbolic inference, pattern matching and tree searching.

CONCLUSION

As powerful as the numeric tools are, there is relatively little we know that can be reasonable expressed as numbers. But the computer is not just a calculator; it is a general purpose symbol manipulator.

The beauty of this technology so far is that is seems appropriate to all industries. The leaders have been in finance, manufacturing and insurance but everyone is participating.

Hypothesis driven, or goal-oriented programming is a means of implementing logic programming using the Prolog language. The whole purpose is to program at a higher level.

Because you tell the computer what you know about the problem rather than how to solve the problem, you can achieve vast improvements in productivity. Programs make sense and can be proven correct so they take less time to write, test and debug.

Most importantly, you are capturing the knowledge about the problem. *In the Knowledge Lies the Power.* The programs work because of the base of knowledge about the case at hand.

Maybe the real computer revolution is yet to come - to use the computer to do reasoning. That is what AI and the much talked about 5th generation is all about.

APPENDIX A

THE HISTORY OF PROLOG

Prolog's the two founders Robert Kowalski (Edinburgh) and Alain Colmerauer (Marseille) worked on similar ideas during the early 70's and even worked together during one summer. The results were the formulation of the logic programming philosophy and computation model by Robert Kowalski (1974), and the design and implementation of the first logic programming language, Prolog, by Alain Collmerauer and his colleagues (1973).

A more potent force behind the realization that logic can be the basis of a practical programming language has been the development of efficient implementation techniques, as pioneered by Warren (1977) with the Prolog-10 compiler.

In spite of all the theoretical work and exciting ideas, the logic programming approach seemed unrealistic. The main claim was that the languages such as Prolog were hopelessly inefficient and difficult to control and could not prove a substitute for Lisp. In the mid to late 70's the Prolog-10 compiler was developed and dispelled all the myths about the impracticality of logic programming. That compiler delivered performance comparable to the best Lisp systems available at the time. Furthermore, the compiler itself was written almost entirely in Prolog, suggesting that classical programming tasks, not just sophisticated AI applications, can benefit from the power of logic programming.

No doubt logic programming would have remained a fringe activity in computer science for quite a while longer were it not for the announcement of the Japanese Fifth Generation Project in October, 1981.

The syntax of Prolog stems from the clausal form of logic due to Kowalski (1974). Warren adapted Marseille Prolog for DEC-10 and many systems adopted most of the conventions of Prolog-10 which has become known more generically as Edinburgh Prolog. Its essential features are described in the widespread primer on Prolog (Clocksin and Mellish, 1984).

APPENDIX B

KNOWLEDGE ACQUISITION

We train people to have expertness but not expertise. We call people such as lawyers and engineers para-professionals - those who master sequences of activity but lack the generative capability for creating those sequences that true experts have. In a knowledge-based system we have to be certain that we are getting the expertise underlying the performance.

Experts guess a lot. They are effective at estimating things in ways that less expert people cannot begin to comprehend. They work at hard problems, get things right most of the time and do that quickly.

But expertise is not just lots of experience. It involves the use that experience to recognize key issues and ignore irrelevant ones. Human experts use *macros* that appear to be guesses. They cannot give IF/THEN rules or cause and effect rules directly. We expect them to reason intelligently rather than intuitively. The process tends to be invisible to the expert and they lack self knowledge. This is known as the Knowledge Engineering Paradox.

Human beings have developed the capacity to recognize and store away the things we do often to long term memory, like compiled programs, so we can run them in short term memory with far less expenditure of resources. The problem is that we no longer know what they contain.

Experts look for the shortest distance from problem to solution. Past experiences provide the shortest path. If not, novel problems can be a good technique to get at the underlying principles analytically using deductive steps.

Heuristics are used to avoid "garden path" results. Experts have *meta-rules* that say "be careful at this point".

Experts can gain a greater awareness of their knowledge by working with a Knowledge Engineer. Knowledge Engineering can create new knowledge by illuminating tasks where experts made mistakes or overlooked important things.

BIBLIOGRAPHY

Baxter, Lewis D. Computer Programming Management. *MPROLOG*. 1986.

Bratko, Ivan. *Prolog Programming for Artificial Intelligence.* Menlo Park, California: Addison-Wesley, 1987.

Engle, Steven; Hodgson, Jonathan; and Vits, James. Death by 1,000 cuts: Prolog on the PC. *Computer Language*, July 1987.

Malpas, John. *Prolog: a relational language and its applications.* Englewood Cliffs, New Jersey: Prentice-Hall, 1987.

Mathews, M. Haytham. Prolog and C. *Computer Language*. July 1987.

Sterling, Leon, and Shapiro, Ehud. *The Art of Prolog.* Cambridge, Massachusetts: The MIT Press, 1897.

Stevens, Lawrence. *Artificial Intelligence, the search for the perfect machine.* Hasbrouck Heights, New Jersey: Hayden, 1985.

DE-MYSTIFYING DATA BASE NORMALIZATION

Patrick C. Witiw and Gil Harrison

Brant Computer Services Limited
9637A-45 Avenue
Edmonton, Alberta
Canada

(403) 438-9123

Preface

On a beautiful autumn day in Edmonton in October 1985, two consultants from a computer services company spoke to a client's analysts and programmers on the subject of the performance of a certain 4th GL upon IMAGE data bases on the HP3000 Series 48.

Following a presentation on the internals of and the interaction between MPE, IMAGE and the 4th GL a pre-lunch question from the audience was asked that the speakers were not prepared for:

> "What effect does the normalization of a base to the 3rd normal form have on performance compared to that of the 1st and 2nd normal form databases?"

The speakers were only barely familiar with the term data base normalization and had to admit, with some embarassment, that they had no working knowledge of the associated techniques. They also made a mental note to investigate normalization soon.

During the next two years, the consultants were involved with the design, implementation and maintenance of a number of IMAGE data bases but never got around to learning about and applying data base normalization.

Finally, the time has come for my collegue Patrick C. Witiw and I to set forth and de-mystify data base normalization.

Gil Harrison

Introduction

Data administration today uses a structured approach to managing a company's resources. The decision makers have access to information, when needed. This is accomplished by combining the efforts of both the data processing people and the end users to identify, define and implement the data bases which contain the company's whole data resource. In doing this the two groups establish the overall logic for the company's data infrastructure. Data bases are now designed and implemented as common data bases and are used by MIS built systems and end user built systems.

There are several major design techniques available for building data bases and this paper will serve to explain one technique call data normalization which is a popular "buzzword" in the vocabulary of data base designers and analysts. At times it seems these specialists use the term normalization to justify their existence to the less informed among us. Though normalization of data is derived from theoretical mathematics and is in the repertoire of the skilled data base designer, we will demonstrate how normalization is a technique by which anyone who understands the functional aspects of one's data can produce data structures that rival those produced by the elite of the data analysis profession.

What is normalization and when do we use it?

Normalization is a technique for decomposing data into smaller structures in which each field is totally dependent upon the primary key of the entity in which it resides. This theory was designed by E. F. Codd who is recognized as the father of data base normalization. Codd translated the origin of normalization theory from abstract-theoretical mathematics into a process which is more human.

An important factor to note about data normalization is that it is not new! Data Base Administrators (DBA) have been intuitively normalizing data for years. The only thing new about the subject is that we are starting to realize that normalization should not be an exclusive skill of the DBA. R. C. Perkinson, a 1980's data base structure and design guru, reinforces this by writing:

> "Normalization depends on a knowledge and understanding of the data in the functional business unit being examined

> and the way it relates together. It does not depend upon any particular data base knowledge or skill." (1)

If you are an analyst, or an end user, or even in a position of management and you understand the flow of your data, then by using Codd's process step by step, you should be able to produce a normalized data structure comparable to one produced by an experienced DBA. And, if I were to ask you your reasons for your entity structure and then ask the DBA the same question I would receive two completely different answers. The DBA would say "I based it on experience". You would give me definate reasons for your structure because you understand how the data is used and you would use normalization to express your understanding. Gane and Sarson, a husband and wife team who specialize in data base design, submit that the normalization process is "inspired common sense".

Without looking very hard we can see normalization as it is applied in our day to day lives. Take, for example, McDonald's restaurants. How successful would the organization be if all they offered was beef, chicken, bacon, fish, potatoes and eggs mixed all together in one combination called McStew? Obviously they would not be where they are today because:

1) The prospect of all the food mixed together is unappealing and would not go over very well with their major end user, the junk food addict,

2) The prospect for accessing what you want quickly is quashed when you have to spend valuable time separating the fries you have craved from the rest of the goo,

3) Too much money would be wasted by the end user eating only the craved fries and throwing the the rest of the goo away.

How about the way you organize your clothes dresser. Do you have your socks, undergarments and sweaters mixed all together through out your drawers? Or do you have them separated in their own respective drawer for quick and easy accessibility? Starting to get the picture? The list of "inspired common sense" is endless. Normalization is the simple concept of breaking down large views of data into simple structures (SIC).

So what exactly is the normalization process?

Here is how it sounds when the DBAs talk: There are three main steps to normalization and each step has its own specific rule. The three steps are:

1) FIRST NORMAL FORM
   - repeating groups are moved into a new entity

2) SECOND NORMAL FORM
   - attributes that are wholly dependent on only part of a primary key or primary compound key are moved into a new entity

3) THIRD NORMAL FORM
   - attributes wholly dependent upon another key within an entity are moved into a new entity

One major problem in developing data bases using the normalization technique is understanding the lexicon of normalization and mathematical terms. We eliminate this problem by eliminating the jargon and replacing it with common everyday terms. However, here are some terms that we cannot avoid and therefore must be defined before we can continue:

ENTITY
- a record containing one or more data fields

KEY
- a data field or combination of data fields used to identify and locate a record

PRIMARY KEY
- a key that is used to uniquely identify a record (eg: a TELEPHONE NUMBER is unique to each household)

SECONDARY KEY
- a key that does not uniquely identify a record; that is, more than one record can have the same key value (eg: an AREA CODE is related to many household TELEPHONE NUMBERs)

During our research of normalization we became quite familiar with the Customer Order Processing (COP) application. This was due to the fact that all the authors of our research material (Perkinson, Martin, Gane and Sarson) used the COP system as an aid for their interpretation and explanation of the normalization process. We, on the other hand, are going to illustrate normalization by examining the development of a Tape Archive and Retrieval System (TARS). TARS is a functioning system that we developed to run on the HP3000, in early 1987. The system was originally programmed using COBOLII but later was converted into a PowerHouse application. The TARS data base makes use of IMAGE, Hewlett-Packard's hierarchical data base management system.

When we first developed the TARS application we used our experience in IMAGE design to build the data base. At the time we did not have a clear understanding of the normalization process. It was only after our research for this paper that we decided to apply the normalization techniques to TARS. We wanted to prove how easy normalization could be.

In doing this our first task was to break down and reorganize the TARS data base fields in order to produce on paper a list which represented the "un-normalized state". We then proceeded to apply the three rules of normalization to the list. The result was the creation of a normalized data base which looked almost identical to the original data base. The differences will be explained later.

Before starting the normalizing process we must develop an understanding of how a user will use the TARS information. In order to gain this understanding it is best that we look at the manual tape archive and retrieval system which was replaced by TARS.

By making use of a file drawer containing index cards called SET CARDS, the manual tape archive and retrieval system provided a Computer Operations department with the ability to keep an orderly account of all magnetic tapes. On these SET CARDS we wrote information pertaining to each set of tapes in our library. There was one SET CARD per tape set. A unique ACCOUNT NAME was assigned to each user client and inhouse department. We catalogued the SET CARDs by account. When an operator was asked to locate a set of tapes he would ask the requesting party for the ACCOUNT NAME to which the tapes belonged, the DESCRIPTION of the tape set's contents and the tape set creation date. Given this information the operator would review all the SET CARDS in the file drawer for the given ACCOUNT NAME. After finding the appropriate SET CARD the operator would use the information on the card to locate the requested tape set. All tapes belonging to a given tape set are stored together on a rack or in a box with the SET NUMBER prominently displayed. The tape number on the band of the tapes would correspond to the VOLUME NUMBERs listed on the SET CARD. Figure 1A is an example of a SET CARD which was used in our manual tape archive and retrieval system.

SET CARD

SET NUMBER 0134
ACCOUNT NAME ACRC
ACCOUNT DESCRIPTION ACCOUNTS RECEIVABLE
SET DESCRIPTION
5/70 MARCH 1987 Version of the INTEREST data base

SET STORAGE Brant Edm. Tape rack # 4
SET SEQ/TOT 4
SET CREATION DATE Jan. 14/86
SET EXPIRY DATE Jan. 14/88

VOLUME INFORMATION

VOLUME NO	LENGTH	SEQ/NUM	INIT/DT	CLEAN/DT	NUM/CL	MANUF
BE1012	2400	1	Jan. 14/86		0	3m Black Watch
BE1013	2400	2	Jan. 14/86		0	3m Black Watch
BCA659	2400	4	Sept. 4/83	June 4/85	1	Memorex
AE101	3600	3			0	Memorex

FIGURE 1A

A small to medium sized operations department can have in circulation anywhere from 500 to 1500 tapes. This is a significant amount of tapes especially if you are the computer operator who is given the task of manually locating a set of archived audit tapes requested by the Accounts Receivable department. We developed TARS as a solution to simplify manual tasks of this nature. TARS is an an online application and will help you to easily and readily find a requested set of tapes. The requested information can be presented to the user in either display terminal format or paper report format. Access to TARS information is done via one of three keys as follows.

1) A unique SET NUMBER represents a specific sequence of related tapes,

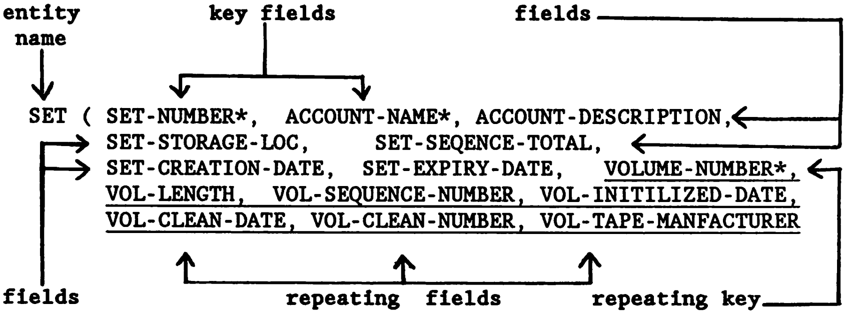
2) A unique VOLUME NUMBER represents one magnetic tape,

3) An ACCOUNT NAME represents a group of related SET NUMBERS.

Now that we understand the function of the data we can proceed to normalize it.

Initial Process

You will recall that a key field is used to identify and locate records. Having determined the keys, we can now create the preliminary order list which will contain all the fields on the SET CARD. The preliminary list represents the "unnormalized state" and the order in which the data is listed is important for ease of analysis. List the entity name first, the keys next, then list the rest of the fields. "*" denotes key.



SET (SET-NUMBER*, ACCOUNT-NAME*, ACCOUNT-DESCRIPTION,
SET-STORAGE-LOC, SET-SEQENCE-TOTAL,
SET-CREATION-DATE, SET-EXPIRY-DATE, VOLUME-NUMBER*,
VOL-LENGTH, VOL-SEQUENCE-NUMBER, VOL-INITILIZED-DATE,
VOL-CLEAN-DATE, VOL-CLEAN-NUMBER, VOL-TAPE-MANFACTURER

Fields that can take on more than one value occurence, such as VOLUME-NUMBER and the information pertaining to VOLUME-NUMBER should be underlined. VOLUME-NUMBER is now known as a repeating key. We are now ready to start the normalization of the SET CARD data.

First Normal Form

As indicated earlier, this step requires us to identify and remove the repeating groups of fields that exist in the entity called SET. This is an easy task since we already indicated the repeating groups by underlining them. So we move VOLUME-NUMBER, VOL-LENGTH, VOL-SEQUENCE-NUMBER, VOL-INITIALIZED-DATE, VOL-CLEANING-DATE, VOL-CLEAN-NUMBER AND VOL-TAPE-MANUFACTURER from the SET entity and into a new entity called SET-MEMBER.

```
              primary        secondary
                key             key
                 ↓               ↓
      SET   ( SET-NUMBER*,  ACCOUNT-NAME*,  ACCOUNT-DESCRIPTION,
              SET-DESCRIPTION, SET-STORAGE-LOC,
              SET-SEQUENCE-TOTAL, SET-CREATION-DATE,
              SET-EXPIRY-DATE )

    new                  compound
   entity                   key
     ↓                ┌──────┴───────┐
 SET-MEMBER ( SET-NUMBER*,  VOLUME-NUMBER*, VOL-LENGTH
              VOL-SEQUENCE-NUMBER,  VOL-INITIALIZED-DATE,
              VOL-CLEAN-DATE,  VOL-CLEAN-NUMBER,
              VOL-TAPE-MANUFACTURER )
```

Removing repeating groups always results in the newly created entity(s) having a primary compound key. In this application the entity SET-MEMBER's primary compound key is SET-NUMBER + VOLUME-NUMBER. The key SET-NUMBER is a unique key in the entity SET and will now be referred to as a primary key. Whereas ACCOUNT-NAME which is not a unique key in the entity SET will be referred to as a secondary key. Labeling keys, as you see, requires an understanding of the data.

Second Normal Form

This step requires us to identify and remove the fields which are wholly dependent on part of the primary key or primary compound key. In the entity SET-MEMBER all of the non-key fields are wholly dependent on VOLUME-NUMBER and VOLUME-NUMBER is part of the primary compound key. So we copy VOLUME-NUMBER and move VOL-LENGTH, VOL-SEQUENCE-NUMBER, VOL-INITIALZED-DATE, VOL-CLEAN-DATE, VOL-CLEAN-NUMBER AND VOL-TAPE-MANUFACTURER into a new entity called VOLUME shown as follows:

SET (SET-NUMBER*, ACCOUNT-NAME*, ACCOUNT-DESCRIPTION,
SET-DECRIPTION, SET-STORAGE-LOC, SET-SEQUENCE-TOTAL,
SET-CREATION-DATE, SET-EXPIRY-DATE)

SET-MEMBER (SET-NUMBER*, VOLUME-NUMBER*)

new entity ↓ **primary key** ↓

VOLUME (VOLUME-NUMBER*, VOL-LENGTH, VOL-SEQUENCE-NUMBER,
VOL-INITIALIZE-DATE, VOL-CLEAN-DATE,
VOL-CLEAN-NUMBER, VOL-TAPE-MANUFACTURER)

VOLUME-NUMBER becomes the primary key of the entity VOLUME. The entity SET does not contain a primary compound key but rather a primary key which is SET-NUMBER. If SET-NUMBER was made up of two sub-fields whereby the first sub-field was composed of two digits which represented a company department code and there were fields in the entity SET which were dependent on the department code then we would perform the second normal form analysis on this entity. However, in the TARS application SET-NUMBER is used as a single field and no dependencies exist on only part of the SET-NUMBER key, therefore the second normal form is not performed on the entity SET.

Third Normal Form

Our last step in the normalization process is to identify and remove the fields which are wholly dependent upon another key or field within the entity in which they reside. The difference between the second normal form and the third normal form is the second normal form deals with dependancies only on primary keys whereas the third normal form deals with dependencies on all keys and fields except primary keys. If an entity in the second normal form has all non-key fields wholly dependent on only the primary key or primary compound key it is then considered to be in the third normal form. The third normal form process is one in which we remove fields seemingly in the "wrong" entity.

Returning to our example, we recognize that the entity SET has a key which is not a primary key or primary compound key. A perfect target for the third normal form analysis. With further examination we see that the non-key field ACCOUNT-DESCRIPTION is wholly dependent on the secondary key ACCOUNT-NAME. As a result, we copy the secondary key ACCOUNT-NAME and move ACCOUNT-DESCRIPTION into a new entity called ACCOUNT shown as follows:

```
       SET ( SET-NUMBER*, ACCOUNT-NAME*, SET-DESCRIPTION,
             SET-STORAGE-LOC, SET-SEQUENCE-TOTAL,
             SET-CREATION-DATE, SET-EXPIRY-DATE )

    new        primary
   entity        key
     ↓            ↓
   ACCOUNT ( ACCOUNT-NAME*, ACCOUNT-DESCRIPTION )

SET-MEMBER ( SET-NUMBER*, VOLUME-NUMBER* )

    VOLUME ( VOLUME-NUMBER*, VOL-LENGTH, VOL-SEQUNCE-NUM,
             VOL-INIT-DATE, VOL-CLEAN-DATE, VOL-CLEAN-NUMBER
             VOL-TAPE-MANUFACTURER )
```

The entity VOLUME is an example of an entity in the second normal form and having all of its non-key fields wholly dependent on only the primary key, thus qualifying it as a third normal form entity.

The result of the third normal form process is to produce entities whose non-key fields are independent from each other but still dependent on the primary key or primary compound key of the entity in which they reside.

We have just completed what we set out to do! With relative ease we systematically normalized to the third form the TARS application data. To restate our point we ask: "what's the big deal about normalization?" A blunt question maybe - but it is necessary to realize that normalization is a simple technique of structured intuition which helps us to organize data base data.

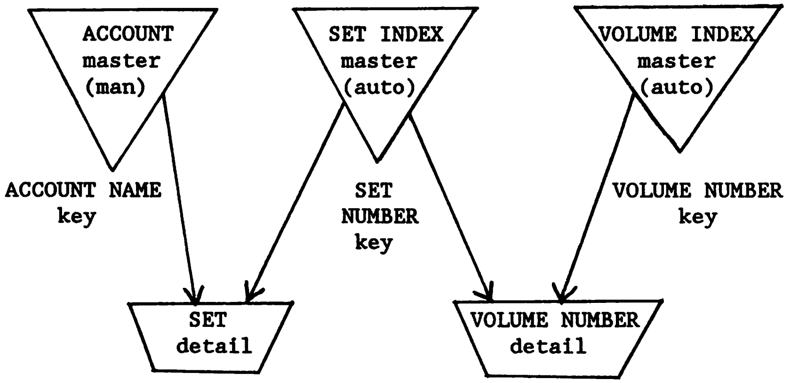
Now we are ready to implement the logical design that our normalization produced. The physical enviroment for our data base is defined by the HP3000 product called IMAGE. It is important to realize that in moving our logical design to the physical environment of IMAGE the entities will be represented as data sets.

The entity SET will be represented by a detail data set. The entries (records) of the set will be accessible on SET-NUMBER and ACCOUNT-NAME via the IMAGE paths defined by the automatic master called SET-INDEX and the manual master called ACCOUNT.

The entity VOLUME will be represented by a detail data set. The entries of the set will be accessible on the VOLUME-NUMBER and the SET-NUMBER via the IMAGE paths defined by the automatic masters VOLUME-INDEX and SET-INDEX respectively.

The entity ACCOUNT will be represented by the manual master called ACCOUNT in which the entries are accessible by the search key ACCOUNT-NAME. This master will also serve as an access path though the detail called SET.

In our implementation there is no need for the logical design's SET-MEMBER entity because IMAGE access paths on the keys SET-NUMBER and VOLUME-NUMBER are provided though the two index data sets. The physical layout of the TARS data base is shown on the following page.


ACCOUNT
master
(man)
SET INDEX
master
(auto)
VOLUME INDEX
master
(auto)
ACCOUNT NAME
key
SET
NUMBER
key
VOLUME NUMBER
key
SET
detail
VOLUME NUMBER
detail

SET (SET-NUMBER*, ACCOUNT-NAME*, SET-DESCRIPTION,
SET-STORAGE-LOC, SET-SEQUENCE-TOTAL,
SET-CREATION-DATE, SET-EXPIRY-DATE)

ACCOUNT (ACCOUNT-NAME*, ACCOUNT-DESCRIPTION)

VOLUME (VOLUME-NUMBER*, SET-NUMBER*, VOL-LENGTH,
VOL-SEQUENCE-TOTAL, VOL-INITIALZE-DATE,
VOL-CLEAN-DATE, VOL-CLEAN-NUMBER,
VOL-TAPE-MANUFACTURER)

index
data set
↓
SET-INDEX (SET-NUMBER*)

index
data set
↓
VOLUME-INDEX (SET-NUMBER*)

These five data sets are exactly the same as found in the original data base designed without normalization. Through normalization we have proven our original design.

Conclusion

To summarize, normalization is a logical level development methodology which in the grand scheme of database design and implementation is only one piece of the pie. Logical development enables us to determine the relationships between the data fields and the entities in which they exist. The other pieces of the pie consist of the following:

1) The physical level of development expresses the logical relationships in terms of software and machine enviroments

2) The actual implementation of the data base

3) The performance analysis of the data base

The success of a data base application running smoothly and efficently depends largely on the physical design, the responsibility for which is in the domain of the DP professional. However, just because the data base application is considered to be successful by the physical designers does not guarantee approval from the end user. The end users will give their approval only when their needs are fulfilled. Involving them in the logical level of development is a key factor in developing a data structure which will be accepted by them. After all, data management is primarily a business function and the decisions regarding the data can realistically be made only by the people who are the ultimate users.

Footnote

1 - R. C. Perkinson, Data Analysis: The Key to Data Base Design, chapter 4 page 41

Bibliography

1) Gillerson, Mark L. Database Step-By-Step. (John Wiley & Sons, Inc., 1985).

2) Martin, James. Managing the Database Environment. (Prentice-Hall Inc., 1983).

3) Perkinson, Richard C. Data Analysis: The Key to Data Base Design. (Lellesly, Mass: QED Information Science).

4) Turk, Thomas A. Planning and Designing the Database Environment. (Van Nestrand Reinhold Company Inc., 1985).

April, 1988

APPLYING EXPERT SYSTEMS
IN THE COMMERCIAL ENVIRONMENT

Karen Hopmans

Brant Computer Services Limited
2605 Skymark Avenue
Mississauga, Ontario
CANADA L4W 4L4

(416) 238-9790

INTRODUCTION

The premise behind this paper is that Artificial Intelligence in general and Expert Systems in particular can (and do) play an important role in commercial computing. AI is out of the laboratory and the companies who are using it in their mainstream computing are measuring significant benefit. Expert Systems have served to improve the quality of decisions, distribute expertise throughout a corporation, consolidate the knowledge of several specialists and provide intelligent on-line training systems.

In this paper, I will take a pragmatic approach to AI. I will define what Expert Systems are and give some concrete and interesting examples of how they are being used effectively. From there I will look at how you can introduce this technology into your environment - what resources are available to you and how to recognize whether or not your application lends itself to Expert System technology.

Artifical Intelligence has reached a new maturity and we are seeing AI companies form consortia. AI is no longer at the crossroads stage; firms are now using it as a part of the way they do business. In fact, the future of AI in the commercial world is in embedded or "hybrid" applications which combine different generations of software technology to take advantage of the strengths of each.

ARTIFICIAL INTELLIGENCE

AI is a broad-based technology that encompasses work in fields such as natural language processing, vision systems and speech recognition and synthesis. The goal of AI scientists has always been to develop computer programs that in some way "think". It is a big field with a lot of research underway but most of AI's commercial impact so far has been due to Expert Systems - programs that represent in software form the knowledge of human experts.

Until recently the role of computers has been mainly to perform tasks at which human beings are not particularly adept, such as filing and sorting information. But people are good at tasks involving language, concepts and abstract ideas and that is what AI is all about. Artificial Intelligence represents a fundamental change in computing but it is not meant to replace the tools and techniques we currently have in use. AI simply gives us more power to solve the problems people encounter.

EXPERT SYSTEMS

Expert Systems are computer programs that model the reasoning of a specialist in a narrow domain of expertise. At this point I will take a moment to explain the difference between Expert Systems and Knowledge-Based Systems - terms often used interchangeably. By definition, an Expert System is just that - a system that performs better than or at least as well as a human expert. A Knowledge-Based System is a more general purpose term which simply implies that the source of the program's power is a large body of task-specific knowledge. Most of the systems in the field today are truly Knowledge-Based only; that is, they have not and may never achieve an expert level. These systems perform as assistants, colleagues and teachers to novices. As you can see, it is less technically precise to use the term "Expert System" in most cases. In this paper, however, I have decided to use Expert System (the more recognized term) to mean both Expert and Knowledge-Based Systems.

"Expert" Systems do not contain a precise step-by-step formula or algorithm for solving a well-defined problem. Instead, they use a collection of if-then rules, ranging in number from ten to several thousand, that tell a user sitting in front of a terminal what to do at each particular stage of a job. American Express, for example, has developed an Expert System to advise its credit authorizers and a rule in that system may be: "If a credit card has been reported lost and the cardholder has since recovered it and can prove his identity, then approve the charge."

Expert Systems are good at solving complex problems that have no "right" answer or problems with many interacting bits of information that vary from case to case, such as the best mix of components for a big computer system.

One of the industries that has embraced Expert Systems is Financial Services. Companies in this industry tend to deal with a high volume of important transactions that involve judgment and some of these companies would even characterize themselves as "decision making factories". Expert Systems can provide a means to automate that process.

A key strength of Expert Systems is their ability to have self knowledge. A program "knows that it is doing" if it can explain its decisions on demand. This capability is usually referred to as an Explanation Facility. In a rule-based Expert System, the user can ask "Why" the system made a particular decision or recommendation and the system will give a sequential display of which rules acted on which facts to arrive at a certain conclusion.

Expert Systems are characterized by their separation of the Inference Mechanism from the Knowledge Base. Because of that separation, we can provide an explanation facility that simply traces the rules that contributed to a conclusion or decision. Moreover, an Expert System's "knowledge," made up of rules and facts, can be treated like data. We can add to it, change it, and examine it, all without changing the program that uses it. Expert Systems give us far more flexibility in dealing with symbolic data than do traditional technologies. In addition, the rules can be heuristic in nature which allows us to deal more effectively with problems having a large search space.

EXPERT SYSTEM DEVELOPMENT TOOLS

Expert Systems may be developed using conventional languages, one of the AI languages (Lisp or Prolog) or Expert System Shells.

Conventional languages tend to be a poor choice for developing Knowledge-Based applications. Conventional languages do not provide any reasonable form of flexible control structures for rule selection, evaluation and execution. Conventional languages cannot evaluate themselves and cannot treat their rules as data.

As for the AI languages available, Lisp programs, based on lists and emphasizing increased programmer control, are favoured in the US, while Prolog products, which rely on database-like queries and emphasize speed, are employed extensively in Europe and Japan. Specialized computers are often needed to execute Lisp quickly, but these are not needed for Prolog. According to Artificial Intelligence Markets, Prolog will continue to grow in importance. Lisp will evolve into a system manager language; systems management functions will be in Lisp, and they will be layered between the operating system and the applications. Prolog and other languages (including conventional languages) will serve as secondary, application languages which will be called from Lisp.

Expert System Shells look after the complex programming tasks, simplifying the building of an Expert System, thus leaving the builder free to concentrate on the knowledge acquisition process. General purpose shells may prove somewhat restrictive, although they are becoming increasingly more functional all the time.

Some Packaged Expert System Shells are:
- Knowledge Craft (Carnegie Group)
- ART (Inference)
- KEE (IntelliCorp)
- ESE (IBM)
- TWAICE (Logicware/Brant)
- S.1 (Teknowledge)

Many of the AI languages and tools are available on conventional rather than specialized hardware. From the perspective of the more sophisticated PC-based Expert Systems tool vendors and the larger conventional hardware and software vendors, the AI market is just beginning to open up. A report published by Ovum Ltd. concluded that the Expert System industry is now established and forecasts that the market will grow to $1.9 billion by 1992.

EXPERT SYSTEM APPLICATIONS

Some companies, like American Express, are making big investments of time and money in large, top-of-the-line systems. Others, like GM and Ford, are treating Expert Systems as a strategic technology and buying a stake in big vendors of the software.

Du Pont takes another approach, letting the user of the Expert System build it himself with existing hardware and inexpensive development software. Du Pont will save $10 million this year - a 1,500% return on software and labour costs. They have 500 systems in various stages of development, with 100 in routine commercial use.

Digital Equipment Corporation has been involved in AI for a long time and has applied 40 big Expert Systems to nearly every facet of its operation. DEC says its Expert Systems save it more than $25 million a year.

The first commercial Expert System was XCON - DEC's Expert Configurator. It was begun in 1978, fielded in 1981 and is used to configure all DEC computers. It configured about 50,000 orders in 1986. DEC estimates that 99% of the systems XCON specifies are properly configured and this saves the company about $6-8 million each year. DEC's former configuration experts have become XCON's supervisors and they check its configurations.

During its first year of operation, XCON had months when 40% of its orders were rejected by its supervisors. When XCON was fielded, it was a very inexperienced expert; it had only configured a tiny fraction of the orders that it faced during a typical month in the field. Its progress from "new expert" to "seasoned expert" took about a year.

DEC also has Expert Systems to assign fulfilment sites (factories) to line items in the orders, match all new configuration requests to modules or computers that were assembled for cancelled orders, paperwork management, floor loading, inventory management, a work-in-process dispatcher, a system to co-ordinate and drive two robots that deliver the work-in-process items, a system to analyze tape drive failures and more.

American Express invited Inference to build an Expert System to help ordinary authorizers perform as well as experts. AMEX has standard statistical models that essentially look for charges that fall outside the typical credit patterns established for different types of cardholders. Following simple guidelines, these programs approve most transactions automatically. But when the models find an aberration, they will not automatically disallow the transaction. Instead, a "maybe" recommendation is transmitted to a human authorizer located at one of four sites around the USA who must search through sixteen screens of information to make a decision.

Inference helped AMEX turn the authorizers' "rules" into a software program called Authorizer's Assistant. Authorizer's Assistant summarizes pertinent information on a single screen and makes a recommendation. It has cut the time of a transaction by 20% to 30% and has helped the average authorizer reduce bad judgments by about 75%.

The system allows American Express to keep pace with its growing number of transactions without hiring a large number of new authorizers. The system's success has relied on the fact that the Expert System can communicate with the data base. It is not an isolated system. Another key component is that the area is well understood with a comprehensive training manual available.

Ford Motor Company has more than 3000 robots installed today with expectation of 5000 to 7000 by 1990. Their maintenance engineers specializing in robot repair were being overwhelmed so Ford turned to its Robotics Centre to build a prototype diagnostic and repair Expert System. The prototype was completed in six weeks by two people with no previous AI experience. The system contains about 100 rules and takes maintenance personnel step by step through the procedure of diagnosing and fixing robot problems. According to Morgan Whitney, the Centre's director, the rule base probably covers only about 20% of the rules that will ultimately have to be written but this has proven to be sufficient for 60-80% of the problems people encounter.

GM, which owns 11% of Teknowledge, has an installation that enables electric motor designers with six months' experience to emulate the skills of 20-year veterans. This cuts the time for a typical job from two or three days of work spread over a few weeks to half an hour.

McDonnell Douglas Helicopter Co. has a project underway to design software to help maintain four Apache subsystems: avionics, flight controls, fuel and the auxiliary power unit. The intelligent fault locator used historical maintenance data to enable technicians to isolate maintenance problems. The crew chief enters the nature of the problem into the computer, which then takes him through a series of steps that quickly identify the problem. The system takes the guesswork out of maintaining the four subsystems. It reviews the historical data and predicts the cause of the problem and the individual component responsible. In addition, a graphic display highlights the failed component.

Boeing's first system to go into daily use was Case, for Connector Assembly Specification Expert. Case tells skilled workers how to assemble each of the roughly 5,000 multiple electrical connectors on a typical airplane. Before Case, workers used to hunt through 20,000 pages of cross-referenced specifications. Case has reduced the time its engineers require to locate the specifications from 42 minutes to 5. Boeing has another project for the Navy that helps a crewman select the right types of acoustic buoys to throw into the water to locate an enemy submarine. Boeing also has a number of prototypes currently under development.

Campbell Soup Co. has developed Cooker, an Expert System that is also known as "Aldo on a Disk". Aldo Cimino retired in May as Campbell's resident expert on the hydrostatic and rotary cookers that kill bacteria. He was a storehouse of knowledge who was about to leave the plant. Instead of that, he spent about 30 days with some Knowledge Engineers from Texas Instruments over the course of eight months and now, when employees from Campbell's have a problem with their cookers, they type the symptoms into a computer, answer a series of questions and are led to the same diagnosis that Aldo would have made.

SUMMARY OF APPLICATIONS

To summarize these applications:

- XCON is the expert configurator. It contains a huge volume of expertise - more than any one human expert - and DEC benefits from consistent configurations that always have the most up to date information.

- The American Express authorizing advisor does not really save the human authorizers much time. It does, however, ensure consistency and allows people with less experience to quickly become seasoned experts.

- Ford, GM and McDonnell Douglas all have systems that perform diagnostics. They help the user of the system quickly locate the root of the problem so that user can do what he is supposed to do - fix it.

- Boeing's Case system is similar in that it reduces the time that the human user must spend in locating the engineering specifications and it has removed the problem of keeping manuals up-to-date.

- Boeing's Helicopter Display system handles the explosion of data. It supports the pilot's decision making process by displaying all the information he requires but only that information relevant to the problem at hand.

- Cooker, by Campbell's, retained human expertise as a corporate asset. A number of companies are looking at using Expert Systems for the same purpose.

GETTING STARTED

The above examples outline how Expert System technology can be successfully applied in a number of diverse application areas. It is an interesting and exciting technology and it broadens the types of problems that we can solve in our computing environments. But how do you get started?

Probably the most important way to begin is to read some of the better known reference books on the subject. I have listed several in an appendix to this paper. From the readings, you can establish a basic understanding of the technology. Armed with an understanding of how it can be applied, you should begin developing a simple, yet representative application. You can do this by hiring a consulting firm with experience in AI to either build an application for you or work with you in the development of that application. As an alternative, you may want to undertake the whole process yourself by acquiring and learning how to use one of the AI languages or Expert System shells.

It is often best to begin on the PC and you should probably look to the PC for delivery. Your initial task should be to prove the viability of AI in your organization. If you are going to be its champion, you have to ensure that your application has a good chance of success, produces measurable results and can be distributed to your coworkers (perhaps as a demo) so that they can see how AI can be applied in your environment. In a minute we will look at some guidelines that can help you determine if an application is a good candidate for solution by an Expert System.

We often recommend that companies in the early stages undertake to develop a prototype Expert System with the help of a consulting firm. Your staff can benefit from working with experts in the field and they gain experience by actually participating in the development of a "mini" Expert System. We feel that this is an excellent, low cost start-up solution. The education process will not take as long for your development people as it is "hands on" and you are left with a working prototype that, in addition, you can use as a model for further AI work.

Expert System shells, on the other hand, greatly simplify the process of building Expert Systems. However, your people will have to educate themselves about tools and techniques - often a rather costly process, as a whole new learning curve is generally involved. You should evaluate both alternatives based on the people you can dedicate to the task and the time available.

QUALIFYING THE CANDIDATE

Before embarking on an application, you must ensure that the project will likely make a good Expert System. Keep in mind that AI technology is just another tool available to us. Our philosophy at Brant is "don't consider AI to be a solution looking for a problem". Rather it is an alternative way to solve a problem that is not effectively solved using traditional programming tools. Research has told us that only about 20% of the problems we encounter in life are numeric or quantitative. The remaining 80% are symbolic or qualitative. AI does not replace traditional techniques but does add power to them and provides a second dimension and alternative to problem solving.

Following is a list of features that characterize an Expert System and help to qualify whether or not a candidate application may lend itself to Expert System technology. These features are:

- The domain is characterized by the use of expert knowledge, judgment and experience.
- There are recognized experts that solve the problem today.

- Expertise is not or will not be available on a reliable or continuing basis (ie: there is a need to capture the expertise because it is scarce, expensive, dependent on overworked experts or will be less available in the future).

- The task requires the use of heuristics (rules of thumb, strategies, etc). It may require consideration of an extremely large number of possibilities or it may require decisions to be based upon incomplete or uncertain information.

- There is an expert available to work with the project.

- The expert is capable of communicating his knowledge, judgment and experience, and the methods used to apply them to the particular task.

- The task is neither too easy (taking less than a few minutes) nor too difficult (requiring more than a few hours) for the expert.

- The system can be phased into use gracefully: some percentage of incomplete coverage can be tolerated (at least initially), and the determination of whether a sub-problem is covered by the present system is not difficult.

- The task is decomposable, allowing relatively rapid prototyping for a closed, small subset of the complete task; and then slow expansion to the complete task.

- The skill required by the task is taught to novices; thus the task is not unteachable.

- Test cases are available.

Before introducing Expert System technology into your organization, you should go through a checklist of points such as these to ensure that the project has a good chance of succeeding. You should not try to build an Expert System to solve problems which people are unable to solve. Nor should you try to solve problems that are being effectively solved using traditional techniques.

The biggest benefits will be felt if the application lies within a company's core business area where functions are performed by many people with different levels of expertise, where there is a scarcity of experts and you want to attain a higher level of expertise through an application or a training tool.

CONCLUSIONS

Expert Systems, as we have seen, have been applied successfully to commercial applications with substantial payoffs and benefits.

Barbara Sanders, director of AI systems and planning at General Motors suggests starting with small steps. "The larger-scale applications with the big payoff will come from people who developed their confidence by building smaller systems," she says.

The tools and the expertise exist today to help you begin applying Expert Systems in your commercial or technical environment. If you start small and if you follow some simple guidelines you can successfully implement Knowledge-Based systems. That will increase your understanding of how Artificial Intelligence can impact your business and give you the experience to work toward larger projects.

I urge you to consider using this technology. Expert Systems can simplify some of the more difficult problems we are attempting to solve using traditional software technology. If you have an application that applies human expertise to solve problems, consider implementing it as an Expert System. You can start small, solving a subset of the problems, and add more knowledge over time.

APPENDIX

SUGGESTED READINGS

W.R. Clockson & C.S. Mellish, Programming in Prolog (New York: Springer-Verlag Berlin Heidelberg, 1984).

Hayes-Roth, Waterman & Lenat, Building Expert Systems (Menlo Park: Addison-Wesley Publishing Company, 1983).

F. David Peat, Artificial Intelligence: How Machines Think (New York: Bean Publishing Enterprises, 1988).

Donald A. Waterman, A Guide to Expert Systems (Menlo Park: Addison-Wesley Publishing Company, 1985).

April, 1988

FACILITIES MANAGEMENT -- A VIABLE ALTERNATIVE

Jack Neale
Brant Computer Services Limited
2740 Queensview Drive
Ottawa, Ontario CANADA
K2B 8H6 (613) 726-0004

"What skills are required to manage and operate an HP3000 minicomputer? Where do I find people with these skills? And what is the most cost-effective way for me to acquire that level of expertise?" These are the questions facing the purchasing departments and managers of organizations who have or expect to receive an HP3000.

A single HP3000 can now address the market of data processing requirements ranging from a few users on the Micro LX to more than 400 on the series 955. A network of such machines can, of course, expand that significantly. One thing that remains relatively consistent across this wide range is the set of skills necessary to manage the computer system(s). The frequency of demand on those skills changes depending on the processor and the resident applications; however, the basic ability to manage an HP3000 is independent of either of those factors. A system manager who is responsible for a single HP3000 Micro LX requires the same set of skills as the one who manages multiple spectrum series machines. Thus, the amount of time required to be dedicated to those tasks will vary.

The first question was *"What skills are required to manage and operate an HP3000 minicomputer?"* Let us take a look at the requirements of a typical system manager. The four basic categories are System Hardware, System Software, Data Communications and Administration. The environmental considerations are intentionally not included because they are not consistent across the HP3000 spectrum (no pun intended). Including this aspect with the system management functions is referred to as Facilities Management. However, the physical environment can be a topic on its own and is not included in this discussion.

I. System Hardware

The system manager must be intimately familiar with the capabilities and limitations of the hardware. This is essential to provide insight into current problems as well as future expansion. Without this insight, you will be working with a severe blind spot. This is not limited to only HP hardware; knowledge of third party suppliers is equally important to provide a solid foundation from which to make recommendations.

Installing and configuring the hardware are basic skills that are required by the system manager. With all the peripherals that HP provides, some combinations of which are incompatible or unsupported, a thorough understanding of the processor and all of the potential peripherals is important. In addition, the effect of a configuration on performance is equally important. There are many ways to configure the same set of hardware, some of which are better than others in regard to optimization. Again, with third party hardware available, this too provides other alternate solutions.

With full compatibility built into the HP3000 family of minicomputers, hardware upgrades are a continual responsibiity of the system manager. Hewlett-Packard's migration from one series to another is as easy as any in the industry. But, you still require a knowledgeable system manager to lead you in the right direction and to facilitate the process. Experience can minimize or eliminate those situations where an upgrade could pose a problem. Proactive planning goes a long way to avoid prolonged downtime during this process.

System performance is a topic on its own and there are many professionals specializing in this area. Suffice it to say that an experienced system manager with an eye for optimization is worth his weight in gold. Postponing that inevitable upgrade for a year or two by squeezing your existing hardware can save a company a considerable amount of cash as well as provide the opportunity to delay the acquisition until newer technology is available.

Let us not forget another aspect of system management ... recovery from system failures. Again, a person who can recover, minimize the data lost, and get the system up again in the shortest period of time can save your company direct and indirect revenue. A seasoned system manager can react to and solve these problems without any lost effort to bring the system up quickly and smoothly.

All of these components stem from a complete knowledge and understanding of the hardware associated with an HP3000 computer. Timeliness is critical when it comes to most aspects of system management. A system failure is not the time to learn about the hardware. Without a sophisticated level of expertise and a knowledge of who to bring into the picture, a lot of time and many opportunities can pass you by in the meantime.

II. System Software

HP3000's are renowned for their complete set of system software that comes with the hardware. Whereas other vendors rely on third parties to provide database management systems, forms packages and other utilities, Hewlett-Packard provides them when you purchase your system. Although you can find other software that specializes in each of the utilities, the system software is adequate and, in the absence of other packages, can perform the required functions. Third party tools provide easier to use and more complete functions but not all systems are blessed with all of the packages to make system management easier. Innovative use of system software can achieve the same result, so a system manager must be fully conversant with all aspects of the tool kit graciously supplied by HP. You would be amazed at what a resourceful system manager can do without the use of specialized tools. Think of it as a blacksmith forging his own solutions without the help of a machine shop. Time is usually the trade-off but, under certain circumstances, this is acceptable. The point here is that your system manager must be able to fill in the holes between the third party tools with system software.

There are also the more routine aspects of system management that require a certain level of expertise in such aspects as HP3000 file and accounting structure, system security, UDC's, system logging, etc. To set up a system and keep it running smoothly requires a solid understanding of these concepts as well as the tools to manipulate them. You will also find that a limited knowledge of IMAGE, VPLUS, QUERY, etc. will help your system manager work with software development people more effectively. Basic programming skills including logical and analytical thought processes can be considered an extra feature but they can become very valuable depending upon how demanding you are of your system manager.

Installation of software packages is usually straightforward and a competent system manager can take that one step further to provide another perspective in an evaluation of software. What may appear to the end user to be an ideal package may have serious effects on performance or other system issues. An experienced system manager can identify these aspects and assist in a more complete evaluation. The same is true for upgrades to operating systems and third party software. The repercussions of blindly proceeding can be catastrophic. Again, a system manager can help to prevent such occurrences and can assist in correcting any temporary shortcomings. Your system manager should be your focal point for any software installations and upgrades to liaise between the end user and the supplier. A qualified system manager has an uncanny ability to understand both representatives and to sort out any misconceptions on either part.

III. Data Communications

Depending upon your environment, data communications can be a never-ending source of despair for any system manager. Continual troubleshooting is a fact of life and a very time-consuming process. In addition to the unreliable performance of most networks are the continual advances in technology. Keeping up-to-date with new releases of data communications equipment is a difficult chore. It doesn't require a very sophisticated network to require the full time dedication of a data communications specialist in addition to your system manager. Planning for the future is increasingly difficult as product announcements are frequent in a very dynamic field. Again, a basic understanding of data communication protocol is essential for effective system management. It is a difficult area in which to become an expert unless you can devote all of your time to that one aspect. This is a perfect example of an area in which a system manager may choose to establish contacts in the industry to assist in his communications network. This ability of a system manager to know whom to contact to resolve what issues is another important trait. Recognizing that he cannot be all things to all people in a field as large as data processing in the HP3000 arena is critical to the successful management of your system. Developing a network of specialists in each field outside of your organization is an effective method of addressing those issues that require very specific knowledge. A system manager must be able to develop those relationships with other people to assist in resolving problems that involve other suppliers.

Because data communications at this point is not an exact science, your system manager frequently finds himself in trial-and-error situations. Making terminal A with slave printer B access port C through a channel on a data switch D over a dedicated line E complete with multiplexors and modems can require some testing. Specifications for cables and configurations are not always available and some clever thinking is usually a necessity. These scenarios can bring into play many aspects of data communications and, again, there is no replacement for a system manager who can rely on previous experience, other contacts in the industry, and an analytical mind to find the solution. He must be able to break down a big problem into many components. Each aspect can then be set up and tested independently so that problems can be identified and resolved in small pieces. Decomposing large problems into many smaller ones is a necessary ability of a system manager.

IV. Administration

Performing the above three aspect of system management without the corresponding administration can cause serious side effects. Although the immediate needs will be met, the long term stability of the data processing environment depends on the system manager correctly and completely administering each aspect.

Production procedures and schedules are critical to the effective management of your system. The system manager must be able to document fully these procedures, complete with recovery instructions. Many organizations depend on the results of the previous evening's processing to perform their current workload.

User training of the HP3000 independent of the software application that they are using is another function provided by the system manager. The user must have a general understanding of the computer environment to appreciate its capability. Simple aspects like sessions/jobs, output priorities, users and passwords, etc., give the user the basic knowledge necessary to make the best use of the system.

Part of the service that your system manager should provide is some level of responsibility for ordering consummables such as paper, ribbons and toner cartridges. He has the best idea of when supplies are running low and you don't want to be missing an important report because you ran out of paper.

If your requirements dictate the need for other operations staff such as console operators, your system manager will have to train and manage these people. On-call support could also become part of the set of responsibilities -- again depending upon your production environment.

For any computer environment, there are contracts for maintenance, upgrades, trade-ins, etc., and your system manager should be the administrator. Establishing the level of support required and the sources for providing such service is the responsibility of your system manager. He has to work within the constraints of those contracts and should be involved in their negotiation.

As your computer system grows, you will require an inventory control to monitor upgrades, purchases and sales of equipment. Your system manager should be involved in these processes and should be responsible for providing the necessary information to the accounting people in your company.

Other duties that a system manager should perform include:

- initiating backup routines including offsite storage
- establishing regular meetings with user groups concerning current status and future planning
- providing technical support in proposals if you are in a service industry
- scheduling and assisting in preventive maintenance
- establishing a tape library and corresponding procedures
- attending user group meetings and reporting back any significant results.

The underlying skills that you are looking for in a system manager boil down to:

- complete and thorough understanding of all aspects involving the hardware, software and data communications of your HP3000 system(s)
- strong written and oral communications skills to work with a wide variety of people from end users to local management to third party suppliers
- effective analytical skills to work through a wide range of situations in a systematic and logical approach
- efficient time management skills to assist in working on many problems at the same time in a productive way.

The second question was *"Where do I find people with these skills?"*. There are three basic alternatives to this problem. The first alternative is to rely on consultants as and when required. The second is to have your own employee fill this position. You can either develop this person from a current employee or hire someone to fill this position. In either case, you end up at some point with a dedicated system manager as a permanent employee of your organization. The third alternative is to rely on a service bureau that specializes in the management of HP3000 systems. The biggest advantage of this solution is that you can effectively have the benefit of a dedicated system manager without the cost or responsibility of managing them.

The third question was *"And what is the most cost-effective way for me to acquire that level of expertise?"*. The first alternative is the best if your demand for a system manager is minimal. However, most system management functions require immediate action and the time required to find an available consultant may be prohibitive. The second solution is best if you require a system manager on a full-time basis. The third alternative is best if you require immediate access to a fully qualified system manager but not on a full-time basis. Most HP3000 system managers are capable of managing many systems concurrently. A service bureau can employ a full-time system manager and divide his expertise and the corresponding cost for that service over many systems. This reduces the cost of having such a person available without compromising the level of service. This does, however, add another component to your computer facility. The system would have to be at the service bureau's site such that the system manager can manage many systems in the same location. This component could introduce additional costs such as data communications equipment but would also reduce the environmental costs compared to a local installation for the same economy of scale reasons as the system manager.

Before deciding which approach would work best for your organization, first establish your requirements in terms of the amount of dedicated system management time that you require. Once you have determined that and if a service bureau is a possibility, then ask them to outline the costs and benefits of providing such a service. There are other factors to consider in your decision and these factors will influence the route that you take. However, the key is to start with a fully qualified professional HP3000 system manager with the skills listed above. Beginning with this requirement will guarantee your success regardless of the option you choose.

POWERHOUSE STANDARDS

Christopher D. Brayman
Brant Computer Services Limited
2605 Skymark Avenue, Suite 400
Mississauga, Ontario CANADA L4W 4L5
Telephone: (416) 9790

INTRODUCTION

During the years that COBOL became the standard of third generation languages (3rd GL's), the industry adopted many standards in structured programming and system design methodologies as MIS departments scrambled to protect themselves from the whims of creative, but inconsistent, computer professionals. The "techies" of this language generation were forced to standardize to protect companies' often huge investments in computer systems.

Standards were developed, such as "Structured Warner-Orr", for writing COBOL applications. In addition, much attention was placed on design methodologies resulting in a multitude of textbooks describing the best approach to systems analysis and design. Certain leaders emerged, such as "Yourdon", who managed to capture all components of the system development lifecycle (SDLC) in a fashion we could understand.

Today, we are still struggling to understand and deal with the impact of fourth generation languages (4th GL's) on traditional programming standards. Certainly, 4th GL's have changed and have eliminated many of the problems found in 3rd GL environments. The inherent organization and structure of the POWERHOUSE products, the leading 4th GL on mini-computers world-wide, have reduced significantly the risk that a creative "techie" will build an application that no one else could possibly maintain. However, even with this structure, I have seen some radically different looking POWERHOUSE code over the past five years.

In our company, we have a variety of projects ongoing in different computer languages. As a professional service organization offering a variety of solutions to businesses in the HP environment, we employ the "old" and "new" schools of computer professionals. Applications are being developed or maintained in low-level languages such as ASSEMBLER, in 3rd GL's (COBOL, RPG, FORTRAN, PASCAL), and 4th GL's (POWERHOUSE, SPEEDWARE). Regardless of the level of programming language, one constant remains:

> Standards in program development are necessary to ensure consistancy between programmers. No matter how well structured the language, "techies" will always find a way to impose their creativity and do it differently.

The intent of this technical paper and subsequent presentation at the IUG in Orlando is to discuss some of the internal standards we have adopted in our POWERHOUSE applications on HP3000 MPE boxes. Some of these standards you may agree with, others you may not. We make no claim to know the "best way" since, by their very nature, standards are simply that -- "standards".

I. POWERHOUSE PRODUCTS

POWERHOUSE is a family of fourth generation language products authored by Cognos of Ottawa, Canada. The software is comprised of a dictionary, a report writer (QUIZ), a menu and screen generator (QUICK) and a transaction processor (QTP). In addition to these development products running on a variety of hardware platforms, Cognos offers the following applications: POWERHOUSE Graphics, an end-user Report Writer (The Expert); a financial accounting package (Multiview); a spreadsheet program (Powerplan); and, a development and documentation tool (The Architect).

General Standards

1. *A minimum level of source program documentation is required* for all QUICK, QUIZ, and QTP programs. This includes:

   - name of program (MPE filename, group account)
   - version number
   - name of programmer
   - title of program
   - expanded description
   - dates created and modified
   - explanation of major changes

   NOTE: Use the DESCRIPTION verb for expanded descriptions in QUICK screens.

2. If KSAM records are not very static, the binary trees must be frequently rebuilt which is CPU-intensive. As well, file-locking is handled differently for KSAM files (in comparision to IMAGE)

since locks occur for the file around a write, update, and *read. As such, KSAM files are only used for generic keyword searches.*

3. Security specified at the file and element levels inside IMAGE is restrictive and costly to change as data must be unloaded and reloaded to a rebuilt database. Therefore, file and element level security should be kept as simple as possible. *Apply security using POWERHOUSE application security.*

4. Any incoming data from external systems should go through a set of validation checks prior to entry of the production database. The following steps are typically performed:
   - o run QTP batch edits against data
   - o add good records to production base
   - o report rejected records
   - o add rejected records to a correction database and use QUICK validation screens to modify and edit rejected records
   - o rerun QTP batch edit against data and add corrected records to production base.

   NOTE: We try to encourage corrections at the source from external systems.

5. *The use of USE files inside source programs for handling standard system-wide calculations or global hilite options is encouraged.* The most common example is for screen hilite options maintained in one source file:

   example:

   ```
   HILITE DATA INVERSE UNDERLINE
   HILITE ID INVERSE
   HILITE MESSAGE INVERSE UNDERLINE AUDIBLE
   ```

   In this way, the hilite options for a system of screens can be changed by modifying the one USE file source statements and recompiling all the screens in the system.

   This feature is not a new one. The COPYLIB in COBOL environments provides a similar capability.

6. When constructing DEFINE statements in QUIZ and QTP where the value of the defined item is calculated based on the values of multiple items or expressions, use the "IF...ELSE" structure. If it is based on the value of only one item, use the "CASE...WHEN" form.

7. Benchmark testing has shown that it is typically faster to extract data into subfiles by QUIZ rather than QTP. Therefore, use QUIZ for creating subfiles where possible.

8. In large systems, it can often become difficult to differentiate among database files and elements and other temporary variables, defined expressions, and alias files. Therefore, we use standard prefixes as follows:

   T -- for temporary variables
   D -- for defined variables
   A -- for alias files

QDD Standards

1. Increase the blocking factor on QSCHEMAC's.

2. Include global options for things like date formats at the beginning of QDD source files.

3. Use RELEASE and VERSION verbs to control enhancement releases. This is especially important in large, complex and dynamic applications.

4. Include descriptions for all files.

5. Include descriptions and help messages for all elements.

6. Any logical edits and display functions for elements should be included in the dictionary.

   example:

```
ELEMENT CASH-AMOUNT 9(008)V9(002) &

   HEADING "Cash^Amount"     &
   LABEL "Cash Amount"       &
   SIGN LEADING "-"          &
   PICTURE "^^^,^^^,^^"      &
   FLOAT "$"                 &
   VALUES - 100 TO 100000    &
   HELP "this field represents the cash"
        "amount of order detail transactions
```

7. Use common element names for like items.

   example:

   DESCRIPTION

   rather than always creating unique items with names such as INVOICE-DESC, PRODUCT-DESC, etc.

QUICK Standards

1. Major verbs in QDESIGN should be placed in the following order in source programs:

   SCREEN
   TEMPORARY
   FILE
   TEMPORARY
   DEFINES
   ITEMS
   GLOBAL HILITES (use file)
   TITLES (as they occur)
   FIELDS (as they occur)
   PROCEDURES (recommended order from QUICK manual)

2. The QKGO file establishes a wide range of run time parameters for the QUICK screens. Default values are set for all parameters with the ability to increase or decrease values according to the application requirements. System programmers must analyze the machine enivronment for available memory, input/output limitations and CPU power. *In larger complex POWERHOUSE applications, parameters should be tuned to ensure optimal use of stack, internal buffers and extra data segments.*

   As a general rule, the programmer should analyze each user group in an application for resource requirements and establish separate QKGO files. Different users will access only a few screens and others may access many. Different screens may have very different requirements for work areas in stack.

   Special attention should be given to the following parameters:

   A. Application Lines

      This parameter involves the stacking options of the SCREEN statement, the application lines QKGO parameter and available terminal memory. The parameter controls the number of lines of simulated terminal memory used for stacking QUICK screens.

If a user moves between a small number of screens, stack all the screens on different 24-line blocks of terminal memory.

In large application systems where a user moves unpredictably through screens, map screens according to levels. The master menu is mapped onto application lines 1 to 24. All menus and screens at level two are mapped to application lines 25 through 48. This is repeated to the deepest level in the system.

B. Procedure Code

This parameter sets the number of 256-byte pages reserved for procedure code in the user stack. Procedure code records from the compiled screen file on disc are read when an associated function is requested (i.e. the user types "E" in the ACTION FIELD and QUICK moves the Entry Procedure into stack). Records are moved into this allocated space according to a paging system maintained by QUICK.

If the parameter setting is less than the number of required procedure code records for the activity on the screen (the threshold level), subsequent activities will require at least the threshold level of reads to the screen file on disc. *Therefore, it is essential that user's busiest screen be analyzed for its threshold level and the parameter for procedures code be set to this value.*

C. Rollback Buffer, Secondary Blocks, Segment Size, and Screen Table

These four parameters work together to control a secondary "paging system" maintained by QUICK in extra data segments. The paging system is used to store spillovers from stack when rolling back updates and screen table information. It is much faster to load screens from extra data segments rather than the original compiled screen file on disc. As such, performance can be improved in on-line applications where large "paging systems" can be maintained.

Therefore, if the machine environment has sufficient memory, increase the size of the "paging system", as required by the application. Typically some form of stack monitoring tool will be necessary to analyze this environment.

3. *Use default screen processing where possible.* Avoid unnecessary procedure code that must be paged in and out of stack as part of the procedural code "paging system". Apply editing to fields with verbs such as VALUES and PATTERN directly.

   example:

   When doing a conditional lookup to a reference file, use an associated DISPLAY or SILENT field to perform the conditonal processing with a VALUES verb, this technique could be used rather than writing procedural code with GETS, etc.

   ```
   FIELD CUSTOMER-NO OF PROJECTS &
           LOOKUP ON CUSTOMER-MASTER

   FIELD CUSTOMER-STATUS ID SAME DISPLAY &
           VALUE "A"
   ```

   NOTE: This approach would be appropriate where a custom error messages was deemed unnecessary by the designers, and excessive volumes of procedural code was otherwise required in more essential processing on the screen.

4. *Where the logic of the QUICK screen requires that procedural control over reading and updating of a file must occur, use DESIGNER files.* Avoid declaring files as SECONDARY when none of the fields appear on the screen and accessing the files only need occur under certain circumstances.

5. *Use REFERENCE files only for LOOKUP ON.* If a procedural GET is required, use a DESIGNER file.

6. *Use the DETAIL file for one-to-many relationships on a single screen format,* rather than two screens with the second screen declaring the primary file of first screen as a MASTER, etc.

   DETAIL files avoid the unnecessary loading of a separate screen into the user stack and extra data segments.

7. *Use the ALIAS file in the following circumstances:*

   o accessing a file on a different path or mode
   o changing multiple key vlaues in IMAGE chains

8. Use the DELETE files with caution because of the application implications of automatically deleting detail transactions. As well, these deleted records must be rolledback into the Rollback

Buffer in primary stack, then to extra data segments or temporary disc files as necessary. *Negative performance may occur with large DELETE files.*

9. It is important to understand the field processing cycle within QUICK as well as the use of FIELDTEXT and FIELDVALUE during this cycle. Four QUICK procedures are used to control the cycle in the following order:

   INPUT ----> EDIT ----> PROCESS ----> OUTPUT

   i) Input. This procedure is used to manipulate data before any editing is performed by the EDIT procedure. *Any changes to the user-entered value should be made to FIELDTEXT.*

   ii) EDIT. This procedure is used to perform any additional editing after those performed by field verbs such as VALUES and PATTERN as part of the ACCEPT verb. There are two values associated with the field. The new value entered by the user should be referenced by FIELDTEXT (for character items) or FIELDVALUE (for numeric and date items). *The old value should be referenced by OLDVALUE (fieldname) to address the value in the record buffer.*

   iii) PROCESS. This procedure is used to perform calculations after the newly entered value has been placed in the record buffer. It immediately follows the EDIT procedure. *The value of the field should be referenced by the actual name of the field.*

   iv) OUTPUT. This procedure is used to modify data between storage in the record buffer and output back to the terminal screen. *The screen display is altered by modifying the value of FIELDTEXT.*

10. *Proper qualification with file references using "OF filename" for all fields in FIELD verbs and procedural code items should be done.*

example:

```
FIELD AMOUNT OF CASH-RECEIPTS

PROCEDURE PROCESS AMOUNT OF CASH-RECEIPTS
BEGIN
     IF AMOUNT OF CASH-RECEIPTS > 1000
          THEN BEGIN
          .
          .
          .
          END
END
```

Proper qualifications as described will avoid errors for ambiguous file references when adding new files to the processing of existing screens.

11. *Include PUT verbs for DESIGNER files in the UPDATE procedure* to ensure that the DESIGNER file is rolledback automatically by QUICK, if an error occurs during the PUT to the database. When performing these PUTS outside the UPDATE procedure or when modifying the UPDATE procedure directly, use the STARTLOG and STOPLOG verbs (as QUICK normally does in the default UPDATE procedure) to control IMAGE logging if it has been enabled.

12. Very complex screens using numerous file structures and lots of procedural code for processing and edits will perform more poorly than simple screens. *Therefore, adopt the KIS principle (Keep It Simple) in the design of individual screen programs*. Spread the processing of multiple files over multiple screens where possible.

13. The design of a QUICK screen hierarchy should balance the logical requirements of the application with the size of the stack required for deep structures. Some logical structures may encourage a very deep hierarchy and conflict with the stack limitations of the machine environment.

    If the logical structure does not demand a deep hierarchy, consider building shallow structures. The savings realized by this design approach can be used to increase work areas in stack and improve system performance.

14. Consider using an external call to a 3rd GL subroutine in the following circumstances:
    i) when on-line edits and/or procedural processing involves very large volumes of procedural code;

ii) where a standard routine is accessed by many users concurrently and requires immediate response.

QUIZ Standards

1. Major verbs in QUIZ should be placed in the following order in source programs:

   ACCESS
   DEFINE
   CHOOSE
   SELECT FILE
   SELECT IF
   SORT
   REPORT
   NOREPORT
   FOOTING (in order of control breaks)
   FINAL FOOTING
   INITIAL HEADING
   HEADING AT
   PAGE HEADING
   (MPE FILE STATEMENTS)
   SET
   BUILD

2. Production reports should be kept in compiled form.

3. Fully qualify all linkages in ACCESS statements and all elements used in the report.

   example:

```
ACCESS EMPLOYEE-MASTER LINK EMPLOYEE-NUMBER TO &
     EMPLOYEE-NUMBER OF TIME-RECORDS
SORT ON EMPLOYEE-NUMBER OF EMPLOYEE-MASTER &
     ON FUNCTION-CODE OF TIME-RECORDS
REPORT EMPLOYEE-NUMBER OF EMPLOYEE-MASTER &
     FUNCTION-CODE OF TIME-RECORDS &
     HOURS-WORKED OF TIME-RECORDS
BUILD
```

4. Wherever possible, use "SELECT file IF" rather than "SELECT IF", since fewer evaluations must be performed to decide if the record complex should be kept or the individual file read.

5. Divide complicated reports into two passes. Extract and select a minimum set of records in the first pass. In the second pass, report information as appropriate.

6. Logical element characteristics such as headings, picture clauses, and floating dollar signs should be included in the dictionary.

7. When using "SELECT IF", specify selection based on items higher up in the ACCESS statement first (primary file first). This allows QUIZ to eliminate records based on partially satisfied selection conditions.

   example:

   ```
   ACCESS EMPLOYEES LINK EMPLOYEE-NUMBER TO &
        EMPLOYEE-NUMBER OF TIME-RECORDS

        SELECT IF EMPLOYEE-STATUS OF EMPLOYEES="A" AND
        FUNCTION-CODE OF TIME-RECORDS = "77"
   ```

   In this way, records from TIME-RECORDS will only be read if the first condition is met; that is, if the Employee Status of Employees is equal to "A". As well, when multiple conditions are on the same file level, they should be placed according to the most likely condition.

8. Always include a NOREPORT verb in QUIZ reports to specify the report contents if no record complexes are reported.

9. Conditional expressions in DEFINE statements should be organized so that the most likely conditions are evaluated first.

QTP Standards

1. Major verbs in QTP should be placed in the following order in source programs:

   ```
   RUN
   GLOBAL TEMPORARY
   REQUEST ONE
   ACCESS
   TEMPORARY
   DEFINE
   CHOOSE
   SELECT FILE
   SELECT IF
   ```

```
SORT
OUTPUT
ITEMS
SUBFILE
MPE FILE STATMENTS
SET
REQUEST TWO
.
.
.
BUILD
```

2. All QTP programs should be tested using a QUIZ program equivalent to ensure the proper understanding of record complexes that are constructed for the OUTPUT phase.

3. Production QTP runs should be kept in compiled form.

4. Fully qualify all linkages in ACCESS statements and all elements used in the run.

5. Conditional expressions in DEFINE or ITEM statements should be organized so that the most likely conditions are evaluated first.

TITLE: System Development Methodologies in the Fourth Generation Environment

AUTHOR: Kimberlie S. Davis

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 0146

TITLE: An Investment for Now and the Future: User Relationships

AUTHOR: Kimberlie S. Davis

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 0147

Data Design Considerations for Distributed Applications

Leigh Solland
Cognos Corporation
2301 E. Lamar Blvd, Suite 416
Arlington, TX 76006

1. Who Cares?

Every company which uses computers is considering, and most are already using, distributed data processing of some sort.

Large companies:

- are connecting remote processors to mainframes
- are decentralizing operational processing
- are building more operation oriented, mission critical systems
- are tying PC's and LAN's together, which have been acquired and developed in isolation
- are standardizing and connecting decentralized processing

Midrange and small companies:

- are building networks of minis, LAN's and PC's.

Why?

- the cost of acquiring and operating mainframes is very high
- the cost of communications for leased lines from mainframes to remote terminals is high and increasing rapidly
- increased redundancy of equipment achieves greater overall uptime, because even if a part of the system is inoperative, the remainder is still working
- reduced redundancy is possible in people, data entry and data, resulting in lower costs, if the distributed system is well designed
- local control is desired wherever local management actually has authority
- hardware and software growth is more modular and more incremental with networks of smaller computers, and therefore not only less expensive but also easier to manage.

existing equipment can often be incorporated into a new network, and used until it is fully depreciated

diverse solutions may be integrated, incorporating turnkey systems, purchased application software packages, and custom development, even when purchased solutions are only available on heterogeneous hardware or software environments

2. What is Distributed Data Processing?

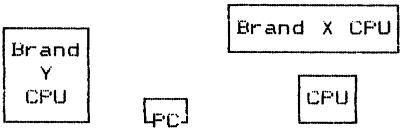
Any or all of the following may be distributed:

- the actual data (storage devices and media)
- the processing (CPU cycles)
- the development effort (programming, report writing)
- control (operational decisions)

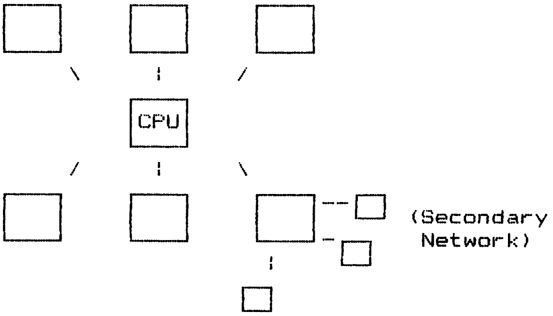
Several configurations are available:

Centralized (one machine, often nearly large enough to perform all processing)

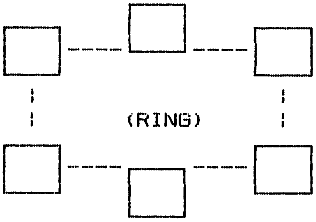
"Distributed" (decentralized authority, with diverse, widely dispersed equipment)



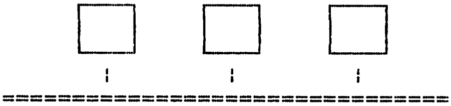
Hierarchical or Vertical (central computer connects to remotes, which may be arranged along geographical, functional or other lines)



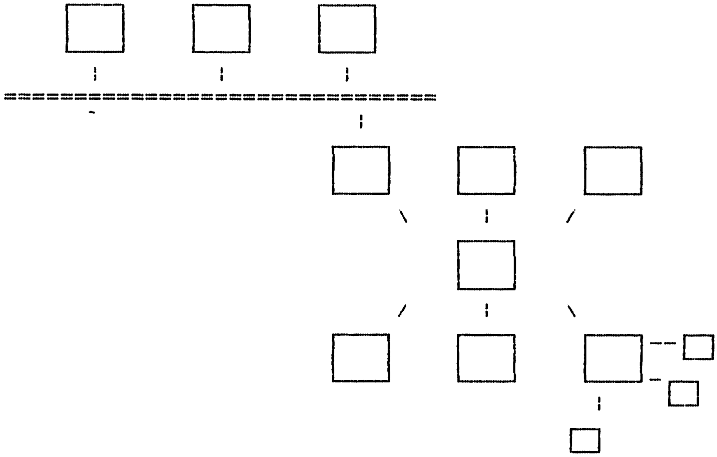
Peer-to-peer or Horizontal (no central computer, but rather a connection which links computers together, such as a LAN)



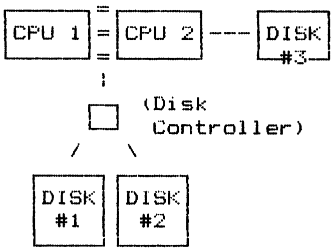
or...



Combinations



Multiple Processors or Clusters (redundancy fosters uptime and reliability, so this configuration is particularly useful at central or otherwise critical nodes)


CPU 1
CPU 2
DISK #3
(Disk Controller)
DISK #1
DISK #2

DDP writing has largely focused on the connection itself, in an attempt to get enough speed and bandwidth to be useful:

- electrical interface (e.g., RS-232)
- asynchronous vs. synchronous communications
- hardware vs. software handshaking
- ISO 7-layer protocols
- DS and NS on the HP3000, other software layers elsewhere
- modem engineering
- LAN's, WAN's and other basic platforms

Less emphasis has been given to the application issues:

- what to distribute (data, processing, development, or control?)
- how to decide?
- how to know when your DDP structure works?!
- how to manage the transition from monolithic or chaotic systems to useful DDP systems?

3. What do we already know about data design?

File design:

one record type per file (or set)

all record items dependent on "the key, the whole key, and nothing but the key, so help me Codd!"

Keys:

fewer keys mean better write performance in Entry/Update

more keys mean better read performance in Inquiry and Reporting, by decreasing expensive serial reads

Locking:

get all locks before you start a transaction

hold the locks for as long, but only as long, as you need them

always get locks in the same order

Tuning goals:

online processing needs to minimize elapsed clock time

batch processing needs to minimize resource consumption

Locking goals:

online processing needs maximum concurrency (tending to shorter transactions)

batch processing needs maximum consistency (tending to longer transactions)

4. What is different in the DDP environment?

Locality issue:

- faster response is possible if the data file is local
- local processing is insulated from down time on other nodes if the data file is local

Integrity issue:

- multiple copies of a file invite unsynchronized changes
- multiple input locations require synchronization processing if multiple copies of a file exist and must be updated immediately
- there is a tradeoff between locking delays, delays for synchronization of update processing, and out of date information

5. There are only THREE possible designs for a distributed file

Centralized or otherwise shared data:

one copy of file, often in central location
shared access to data of common interest

ADVANTAGES:

- easy to update (one place for modifications)
- easy to regulate (locking, duplication, data validation)
- data is always up to date
- fast and easy for analysis and "global" reporting.
- good for reference data which changes frequently(e.g., customer credit or accounts receivable check lists or commodity product prices.)
- good for data updated often from more than one place
- often a very good solution for data which has become "informational", and is no longer of "operational" value.

DISADVANTAGES:

- will always be slower to access and update, if located on a remote node, than local data.

Copied (or "cloned") data:

multiple copies of a file
local access to data of common interest

ADVANTAGES:

- fast access on reads (always local)
- high network reliability (not dependent on any other node)
- must be modified in central location and redistributed periodically, or else all modifications must be propagated to every copy throughout the network
- best for data which is very static, (e.g., lists of credit terms, branches, state tables, zip code lists) or where updates always come from one place (e.g., supermarket price lists).
- could be appropriate for types of data such as customer lists, prices or inventory masters, if they change infrequently.

DISADVANTAGES:

- may be very slow on writes, depending on "write-through" requirements to update copies on other nodes.
- updates are generally difficult, with locking considerations and deadly embraces over multiple CPU's, and dependence on other nodes' uptime.
- may be out of date.

Split data:

local access to data of local interest
this is what many people think of as "distributed" data

ADVANTAGES

- very fast and reliable for local data
- good for local transactions (orders, production) or detail of interest only to local users, but not needed by the entire organization (directions to drive to customers' offices, local inventory balances)

DISADVANTAGES

- slower and potentially less reliable for access to remote data.
- application complexity may increase, because each application has to either incorporate the logic to search for remote data or else use an extremely slow brute force serial search of all nodes.
- it is very difficult to do analysis or reporting on a basis broader than the local versions of the files. It is usually necessary to combine selected records from each node in a single location first.
- the data records are very susceptible to duplication and divergence due to unsynchronized updates.

6. How do you decide which to use?

Who needs to see the data?

- consider that reports may be run at remote nodes and sent to management at headquarters
- Do they need all the detail? Can it be extracted or summed? (For instance, accounting people like to collect data, but do they really need details on which salesman called on which prospect for the past twenty years?

How do data updates happen?

- Who creates or deletes data records?
- Who writes, updates or changes data?
- How often does the information change?
- How soon do the changes need to be available to everyone who has a need to see the data?

How large is the file?

- consider both absolute size (bytes, records) and relative size (compared to other data in the system or organization)

What is it used for?

- customer inquiries?
- monthly G/L batch processing?
- weekly management reports?
- on-demand management reports?
- which manager?

What other constraints must you consider?

- communications costs?
- response time minimum?
- current equipment to be used?
- company standards to be dealt with?

What does your ORGANIZATION look like?

- model the real world, not the current model!

7. What is ideal?

Guidelines:

READ/WRITE LOCATION:

if data is read by one node only, put it there! ... else ...

if data is read by multiple nodes, but written (created, updated, deleted) by one node only, consider locating it at the writing node ... else ...

if data is written by multiple nodes, consider centralizing it if it is dynamic, or cloning it if it is static.

UPDATE FREQUENCY:

if updates are done seldom or never, multiple copies are fine ... else ...

if updates are frequent, one copy is better.

UPDATE DELAYS:

if delays in seeing updates to a shared file are acceptable, look at capturing local update transactions and batch transferring them to a centrally located, single copy of the file ... else ...

if updates must be immediate (e.g., in a sales inventory), the updatable files must be available on-line and performance is much more of a consideration.

FILE SIZE:

if the number of records in the file is small (e.g., a table of state codes and names), it may easily be cloned ... else ...

if the file is large (e.g., the income tax rolls for the U.S.), it will tend to be centralized for storage requirements if for no other reason.

Look at data in a hierarchy which matches the organization

Put each data item at the level where it needs to be, then make it into records. For example, it may be wise to break the Inventory product master into two files, a central or cloned file containing common information such as description and size, and a split file with local information such as stock balance, pricing, and so on.

Do not accept that head office users have a need to see or change anything in the system. If they have such a need legitimately, it may be possible to set them up as valid users on each remote node.

Consider that, to generate common reports, it is probably a good idea to have all the data required by that report at one place. This may be done in batch, if it is only required on a monthly or annual basis.

For senior management reporting or other analysis, snapshot extraction is a very useful option. A periodic process (daily, weekly, monthly, hourly, or ...) creates a new file for the reporting processes, replacing the current reporting file. A snapshot file eliminates old or unneeded records, summarizes wherever possible, eliminates operational necessities such as status flags, and generally loses unnecessary detail. The snapshot process has a side benefit of keeping data consistent throughout a given time period.

Centralize everything to a level which allows it to be updated by those who have a legitimate need to do so.

Intentionally decentralize for performance reasons only.

Plan to copy and download static information periodically.

Large central files, such as Customers, may be considered too large to copy onto each remote machine. These may be split up according to which location deals with them. Then each node can get a copy of "its own" customer list.)

Watch out for bottlenecks:

if all processes on all nodes depend on a name-and-address master file at the central location, the network is very vulnerable to communication or central node failure.

Be creative:

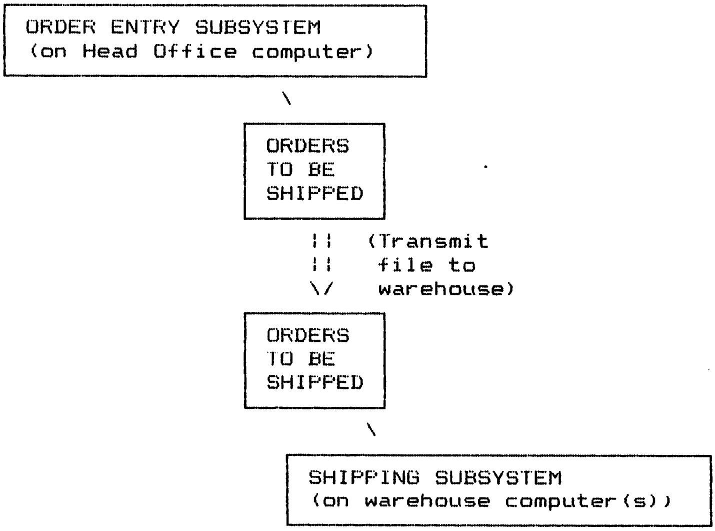
- if one or two data items are messing up your design, break them out into their own record or records, and see what happens!

8. What if the ideal is not possible?

Telecommunications Cost:

Millions of dollars per year may be saved by replacing dedicated leased data lines and online terminals with a network based on dialup. The disadvantage is that the files we so carefully placed in the inner layers of the network may no longer be easily available.

Careful design will separate subsystems, which may be running on completely different machines, with common files (i.e., the output of one subsystem is the input to the next). This makes the network transparent to the application analysts and programmers, so that they do not have to consider it any further.


ORDER ENTRY SUBSYSTEM
(on Head Office computer)
ORDERS TO BE SHIPPED
(Transmit file to warehouse)
ORDERS TO BE SHIPPED
SHIPPING SUBSYSTEM
(on warehouse computer(s))

The communications interface between systems then becomes a matter of transferring a file from one system to a lookalike file on the target system.

This interface may then be purchased, developed, tuned or enhanced in isolation from the applications themselves.

Performance Limits:

Network performance may not be satisfactory, no matter how much is spent, if the data is not local.

use audit files to capture transactions, then batch jobs to propagate them around the network. It may be possible to use a file of nodes which need to be updated for each transaction, so that all nodes don't need to be updated for each transaction.

Head Office or other Reporting

use audit files to capture transactions, and batch them up on a periodic basis. The control of the transfer process may be at the remote node or at the center.

reduce the size of what is transferred as much as possible, by use of extraction and summarization.

Intentional Redundancy Requirements

it may be possible, by cloning each remote node's processes and data in a central location, to increase available up time to the remote users. If the local hardware is down, they may still be able to work, using their terminals and the network to access data and programs on the central hardware. I do not feel that this is a particularly effective way to ensure up time, for several reasons: it takes too much work to keep data synchronized, it introduces a need to update both ways for when a disabled remote node comes back up, and these computers are very reliable in the first place!

9. Summary

Make the network model fit the organizational model

- if the structural philosophy of management changes, your network may also have to change. This is known as "the cost of doing business"!

Concentrate on application functions first.

- where does management think they happen?
- where do they really happen?
- where does management want them to happen?

Look at data next.

- put it where it needs to be to accomplish the functions.
- if you have a problem locating data, review your function map: is a function in the wrong place? (Isn't modelling fun?!!"

Then, and not before, look at hardware, baud rates, muxes, phone lines, and the like.

- it may be necessary to modify the model due to costs, communication service availability, or existing investments
- make your compromises here, but BE SURE that senior management is aware of what is going on: they may decide to spend more, reduce a requirement or change a constraint to achieve the overall business goal. Senior management makes business decisions, technical management makes technical decisions. Each of you should let the other do THEIR job!

Start with a pilot

- don't try to implement the entire universe in one shot, or you are doomed to failure before you start.
- get feedback from the installers, users and maintainers of the pilot before proceeding.
- it is not (quite) too late yet to change your mind.

You will never finish this project!

your model, your network and your applications must continually flex to adjust to the changing business climate, management requirements and available new technology.

PRESENTING TECHNICAL INFORMATION TO MANAGEMENT

George B. Scott
Great Business Solutions, Inc.
P.O. Box 950
Graham, WA 98338-0950

During the past twenty-plus years of working in technical areas, I have had the opportunity to present technical information to management on many occasions, and have been present when others did the same. Hopefully, using some of the guidelines I have developed for myself from my learning-by-fire experiences will help you succeed in achieving approval for your proposals. I have organized these "rules" into the following categories:

1. Knowing your subject.
2. Knowing your audience.
3. Selecting the material to present.
4. Organizing the presentation.
5. Presentations do's and don'ts.
6. Why proposals are accepted/rejected.

A lot of the information contained in this paper may be considered "pretty simple stuff" or to be "obvious". What leads me to believe the contrary is true is the number of times I have witnessed great ideas and opportunities being rejected because of a failure to effectively use some of the following "simple stuff".

I. INTRODUCTION

Generally when you are presenting technical information to management, it is not because management is interested in learning about all the wonderful and neat things you do; usually, it is to gain approval to spend money in some fashion, such as capital acquisitions, additional staff, training, supplies, services, etc. Thus we come to the first and main rule.

Rule 1: Remember that the sole function of your presentation is to gain approval for your request.

All other ideas and concepts which you consider in preparing your presentation must be subservient and compatible with this rule. Straying away from this rule will usually get you in trouble and frequently leads to rejection of the proposal. If you want to stop reading now, just try and follow Rule 1.

II. KNOWING YOUR SUBJECT

When I was a Boy Scout, it was called "Be Prepared". This is such a simple task, especially for the technically oriented person, that you might wonder why it should be mentioned. Very simply put, this is one area in which "Instant Death" can be delivered to your proposal. Over the years, I have seen several proposals rejected in the middle of the proposal because the presenter had failed to adequately prepare himself/herself and could not answer a relevant question.

Knowing your subject is similar to studying for a final examination. You really don't care about all the information; however, you must be prepared to answer any question with a sound answer. Lets look a simple example contrasting the unprepared with the prepared.

You are making a presentation to acquire an uninterruptible power system for the computer room. You know the power requirements of all the computer hardware and have built in a 100% contingency for growth. You are asking for capital to acquire Model XYZ to provide this capability so that you can be on-line to all users the moment power returns to the rest of the plant. All is going well when the following question comes up.

Scenario 1;

Question: "Does this power the air conditioner also?"

Answer: "NO" << Your in trouble, you don't know. >>

Question: "Won't the computer shut down if it overheats?"

Answer: "Yes, but most power failures only last a few seconds."

Question: "Could the air conditioner be added to the uninterruptible power system without overloading it?"

Answer: "I don't know." << Sure Death !!! >>

Scenario 2:

Question: "Does this power the air conditioner also?"

Answer: "Yes. We investigated the additional power requirements to keep the computer room cool for as long as the power was out of service. We found that a simple switch could be installed in the air conditioner which would require only the unit associated with the computer room to have power. By doing this, the power requirements are minimal and fit well within the manufacturer's guidelines. When we go over the detail expenditures, you will notice a cost of modification of the air conditioner which reflects this hookup.

By the way, Sir/Madam, I would like to complement you on a very good question."

Based upon the above scenarios, you can see the effect of being prepared. Put yourself in management's shoes. Which answer gives you the greatest belief that the requestor knows his subject and can successfully complete the project.

Rule 2: Know you subject.

III. KNOWING YOUR AUDIENCE

The second area in which I have seen a great many proposals die is by not communicating effectively. I have seen computer "nuts" start talking about the EGA vs. the VGA cards; the fact that average seek time is 5 milliseconds faster with disc drive B rather than A; the relative speeds of disc cacheing in memory versus disc caching on drives and the associated effect on memory use; and, equally interesting other "techno-garbage".

Of course, if you glance at the officer(s) who must approve this expenditure, you see sort of a glazed look in their eyes, or worse, a look questioning the sanity of the speaker, or still worse, a look of disapproval for wasting his/her time with such a presentation.

The idea here is to forget, forever, any hope you have of "teaching" management. This is in direct conflict with Rule 1. Learn to present your story in terms your audience already understands. This requires you, the presenter, to find out ahead of time something about the background of the key players to whom you will be making your pitch. I have seen instances where ten's of thousands of dollars were spent analyzing a technical situation, more thousands spent in trying to derive the best possible solution, and then watched the entire effort go down the drain because only a very minimal effort was spent in preparing the presentation.

Telling (or Selling) the story effectively to top management is usually far more important to the success of your proposal than knowing with great precision all the facts related to your proposal.

Remember that your job is to improve things and go forward. The single greatest difficulty technically oriented people usually have to overcome is learning that gaining approval for a request is rarely based upon the technical merits of the proposal by itself.

Rule 3: Know your audience.

IV. SELECTING THE MATERIAL TO BE PRESENTED

Perhaps the most difficult facet of presentations I have had to learn over the years is in selecting what material should be presented from a giant sea of "really great stuff".

I try to remember that my objective is not to train the audience to be technically competent in my field. Instead, I select those few facts which directly and dramatically support the picture which I'm trying to paint. Suppose we are trying to request that one million dollars be spent in the next years budget to expand the computing power of the current system. This will include migrating from Series 70 to Series 950 computers, something to which I hope you can relate. Rather than show a series of slides which show the increase in number of users, number of transactions, number of reports, electronic-mail volume increases and other such phenomena, I would rather show a single chart which shows past and projected response time as excess capacity dwindles to zero and an overloaded CPU is achieved. This chart could be compared to the dollars in labor which could be saved by having sufficient computer response to maintain high productivity in computer related activities.

Top management cares about lost productivity when workers sit in front of a terminal doing nothing except waiting on a response from the computer. I would be very surprised, however, if top management would be interested in the fact that the number of packets of data transmission increased from 1.6 to 1.8 millon per day during the last month and show a steady 15% growth per month. The old adage is "KISS" or "Keep It Simple, Stupid".

One of the questions I ask myself is, "Would this hurt my objective if I removed it from the presentation?" Usually, the answer is that the presentation would be more effective without it.

The second thing I have learned in this area (and at a very great expense to me), has been that the more area you cover, the more likely you are to strike someone's hot spot. Regardless of how well you think you know your audience, if you don't restrain yourself in this area, you won't be around long enough to know your audience at all.

Rule 4: KISS

V. ORGANIZING THE PRESENTATION

Several years ago, I had a boss, mentor, to whom I'll be forever grateful. Of the many things he taught me, one was to be up-front when asking for money. If fact, he had a rule that the first sentence of any written request should contain the dollars being requested and the reason for the request. There have been times when I have worked for two days to write the first two lines of a request. They are the most important lines of the request. Often a manager will make his/her decision by the time the first half of a page is read. Rarely will a key decision maker even read the appendices.

Thus you must sell your idea at the very beginning of the presentation. If you turn off your audience or lose them at the start, the odds are that you will never recapture them or win them over to your side. In some proposals, there exists some costs because people will need to be re-trained or reorganizations will be required. Think of this as a game in which the score must always be positive in your favor. First, tell them about the good things you are going to accomplish. After that, the cost required to accomplish these things seems much more palatable.

Rule 5: State the dollar cost and expected accomplishments at the beginning.

Rule 6: Present the positive things you expect to achieve prior to any requirement which could be viewed negatively.

VI. PRESENTATION DO'S AND DON'TS

Although this entire paper is about presentations, there are some really simple things that the uninitiated might not fully appreciate. Over the years, I have seen people fail to get approval, wreck their career, or put themselves in a position of ultimately leaving the company because they failed in one of the following areas. Alternatively, others have received promotions, bonuses and new opportunities because they did it right. This is not meant to be a comprehensive list, but represents only some very key points.

1. Be honest.
2. Be positive and confident.
3. Demonstrate you know your subject.
4. Be receptive when ideas are presented to "improve" your proposal.
5. Be serious, but maintain a sense of humor.
6. Dress appropriately.
7. Speak / write clearly.

In summary of the above,

Rule 7: Maintain an even keel.

VII. WHY PROPOSALS ARE ACCEPTED / REJECTED

I have come to the conclusion that proposals are generally approved because the top management involved believes in the person, not just the idea.

My second conclusion is that most proposals are rejected because of a lack of ability to communicate effectively with management, not because of an incompleteness in research or an unworthy idea.

My third conclusion is that the timing of a proposal is critical. Furthermore, if a proposal is ever rejected because a lack of funds during a period of austerity, it will be much harder to win approval if requested a second time, even if requested in times of prosperity. If you believe that approval is not probable or is questionable, then don't ask.

Rule 8: Time your requests carefully.

VIII. SUMMARY

Your career rides on your ability to sell your ideas to management. Do your homework. Time your proposal carefully. It's a lot easier to get approval for your next proposal when your last request was approved and successfully implemented. When you complete a project on which key management personnel had to give their approval, make sure you inform them of the success and thank them for their assistance and support.

Rule 9: Ring your own bell.

Rule 10: Thank the appropriate people.

I hope the aforementioned guidelines will make it easier for you to get approval for your next proposal.

Comparing UNIX with other systems

Timothy D. Chase
Corporate Computer systems, Inc.
33 West Main Street Holmdel, New Jersey

The original concept behind this article was to make a grand comparison between UNIX and several other well known systems. This was to be all encompassing and packed with vital information summarized in neat charts, tables and graphs. As the work began, the realization settled in that this was not only difficult to do, but would result in a work so boring as to be incomprehensible. The reader, faced with such a wealth of information would be lost at best. Conclusions would be difficult to draw and, in short, the result would be worthless.

After tearfully filling my waste basket with the initial efforts, I regrouped and began by asking myself why would anyone be interested in comparing UNIX with another operating system? There appears to be only two answers. First, one might hope to learn something about UNIX by analogy. If I understand the file system on MPE/V and someone tells me that UNIX is like that except for such and so, then I might be more quickly able to understand UNIX. This I felt was an unlikely motivation. After all, there are much simpler ways to learn UNIX.

Instead, the motivation for comparing UNIX to other systems must come from a need to evaluate UNIX. If we are aware of the features or short comings of other systems, then we can benefit by evaluating UNIX relative to those systems. Choosing an operating system or computer is a major decision which we can benefit from or be stuck with for a long time. So I decided that this article should discuss some of the important UNIX issues in the hope that it might help with decisions regarding the operating system. Naturally such a discussion results in comparisons with other systems, but certainly not in the way I had originally imagined.

So why are people interested in UNIX? Conventional wisdom indicates several reasons. First and foremost is because UNIX is a vendor independent operating system. Many feel that if software development dollars are invested in UNIX, they are safer because UNIX is not tied to any given vendor. If, for example, you write a program in FORTRAN for your HP3000 which makes use of the features of MPE/V, you are going to have to spend some money to move that application to another vendor's hardware. To a large extent, HP has you trapped in that you'll often choose to put up with things that make you unhappy rather than switch to another computer vendor.

The dream of UNIX is that it makes computer hardware more of a commodity than it is now. You can develop UNIX applications then shop for delivery vehicles. This may or may not actually be true, as we shall see, but it most certainly is not true for proprietary operating systems like DEC's VMS or HP's MPE. An excellent model for this approach is the IBM developed PC. Because of the PC's open architecture and generally available operating system, you can now purchase PC's from literally hundreds of hardware vendors. Currently, one can secure a 1 megabyte PC with a color CRT and a 40 megabyte hard disk for under $2,500. It has no nameplate on the front, but it runs Lotus as well as the IBM original. It's the user's dream that UNIX brings this state of affairs to the minicomputer.

The second important feature which UNIX promises is people portability. Nowadays it's not enough to get a COBOL programmer, you need to get a VAX/VMS COBOL programmer. Each proprietary operating system has its own command language, editors, compilers, file system and system services. Only the most superficial of programmers is not effected by the host operating system.

With UNIX, that changes. We have programmers working for us who don't really care what machine they're working on. All they know is that they are working on UNIX. As program managers, this is a powerful incentive to use UNIX. It's especially true for managers of internal service organizations supplying general programming talent. Rather than force the company to standardize on a given vendor or vendor's processor line, UNIX allows you to address a range of processors with a single pool of programming talent. Clearly a cost effective use of an expensive commodity.

A third reason to consider UNIX as an important system is the available software pool. Because UNIX is widely available, it behoves third party software vendors to develop products which are UNIX based. Because UNIX is installed on so many systems, it gives software houses a large potential market. This naturally results in more packages for you to choose from. In addition, competition causes the resulting packages to be high quality. Contrast this with proprietary operating systems. The HP3000 MPE system is actually too small to induce big software suppliers to make the (often extensive) changes necessary to port their packages. The smallness of the potential market associated with proprietary operating systems limits the choices you have and effects the overall quality of available third party packages.

So, our comparison of UNIX with other operating systems is motivated by the hope of evaluating the goodness of UNIX to see if it might be a useful way to gain vendor independence, people portability and access to existing software.

As easy as goodness is to talk about, it is quite another matter to define it. This is because goodness is, of course, relative to how the system is going to be used. It is generally felt that UNIX lacks certain features which would make it a good real time system. Does that make UNIX bad? If you don't care about real time features then it certainly does not effect you. So, although we will evaluate UNIX relative to other systems, our conclusions will only be valid with respect to your scope of application. If you live in Oneonta, New York and there is only an HP office near by, then you may not be really interested in UNIX's vendor independence while a customer in downtown Manhattan might find it very attractive.

Open versus Proprietary

The first UNIX feature to be compared is its openness versus other operating system's proprietary nature. In a nutshell, UNIX's openness stems from the fact that it was provided in source form to most users and is available in some incarnation on many different computers. (For security reasons, UNIX sources are no longer automatically shipped with the system. Now, in order to obtain sources one must purchase a source license.) Written primarily in a machine independent way in the C programming language, UNIX trades off wide availability against somewhat diminished performance. It should be unsurprising that a proprietary operating system executing on the computer it was designed for should out perform UNIX. Because UNIX was written with portability in mind, it cannot take advantage of the special features available on any given computer. UNIX must first abstract the feature and then implement it in such a way that it is available on many different processor designs.

In practise this is so difficult that it borders on impossible. The result is that UNIX has different flavors which address differing underlying hardware capabilities. For example, in VAX/VMS, the operating system is strictly demand page and the hardware is designed to accommodate this. UNIX, however, must cope with machines which can support demand page organization and those which cannot. As a result, UNIX may be run in either paged or swapped mode. This fact causes a difference between individual UNIX installations. Fortunately, the number of application programs which can (or must) differentiate between the two modes is small. Still, the pure concept of a universal UNIX has to give way to questions of efficiency.

Although next to impossible to verify, one confidential study I have seen indicated that a VAX/780 running UNIX could support 32 users while the same machine running VMS could support in excess of 48. The point being not a quantification of the difference in performance, but rather verification of the existence of a difference.

For many applications, however, the ability to execute on different machines unchanged is more important then a slight decrease in performance. But, there are those who don't find UNIX all that pure on the different machines it runs on. A recent issue of Datamation quotes P. J. Plauger as saying "Currently, there are so many dialects [of UNIX] that the idea that there is one UNIX is silly". Plauger, who used to work with Bell Labs during the UNIX genesis appears to know what he's talking about. A partial listing of UNIX system currently in use includes the Sixth, Seventh and Eighth Editions (sometimes called Versions 6, 7 and 8), Programmer's Work Bench, System III, System V, System V Release 2, System V Release 3, Berkeley 4.1, Berkeley 4.2, Berkeley 4.3, PC/IX, UniPlus+, Ultrix, Venix and XENIX as well as a host of UNE-alikes such as Idris (from Plauger's company Whitesmiths) and Coherent.

In addition, various companies like HP, DEC, SUN, Apollo, Plexus ModComp, etc. all offer standard UNIX with a few enhancements just to make it run better. Advertisements offer

"Standard System V with Berkeley BSD 4.2 enhancements". The net result of all of this is that, although UNIX is conceptually open, pure and portable, the local enhancements tend to make it become proprietary in subtle ways. In UNIX, Local enhancements take two different forms. Usually they are commands which have been added and are therefore available only on the enhanced system. Other, braver, users actually modify the resident operating system. This can result in a UNIX which looks normal, but behaves in distinctly abnormal ways.

Though you probably never thought of comparing UNIX with UNIX as another operating system, you should. Take, for example, standard UNIX (whatever that is) compared with HP/UX. HP's real time enhancements (just to make it run better) include the introduction of more than 10 new system calls not found in System V which were either taken from Berkeley 4.2 or invented by HP. If you use these calls in developing your application, then you'll find that UNIX can be just about as bad as a proprietary operating system when it comes to porting to another vendor's hardware.

In the face of all of this, AT&T magnanimously offered to standardize all of the UNIX implementations by introducing the "System V Interface Definition" and an appropriate set of test programs to measure any given implementation's adherence to the standard. The SVID, as it's called, was met with less than enthusiastic acceptance from AT&T's competitors who narrow-mindedly complained that AT&T was actually trying to control the UNIX marketplace. To address this complaint, the IEEE organized an alternative interface standard based on SVID given the project number P1003. This system is called POSIX; a name derived from Portable Operating System. This name is hopefully far enough from UNIX to avoid the legal wrath of AT&T for trademark infringement, but close enough so that everyone (wink, wink) knows what they're talking about.

The conclusion here is a bit cloudy. Although UNIX does represent a giant step toward a portable machine and vendor independent operating system, there is still quite a ways to go. Trivial programs will clearly port without change. More complex applications which require local enhancements to UNIX will not. In point of fact, it would be just about as easy to write the trivial class of programs to port to any operating system be it proprietary or not. As users, we must remember from all of this that when someone says "UNIX" we might need further clarification.

Some bits of history

How did UNIX get to be so varied? Didn't it all come from Bell Labs? In order to answer this, it's worth our time to understand a bit of UNIX history and compare this history with that of other operating systems.

The UNIX time line begins way back in 1965 when Bell Labs was working with MIT and General Electric on Project MAC. The goal of this effort was to develop MULTICS. A large and complex system, MULTICS never totally met its design goals. Reading about MULTICS provides an interesting insight into UNIX's conception. Many of the features one might credit to UNIX actually came from project Mac.

Bell left Project MAC in 1969 and one of the team members, Ken Thompson, started working on an operating system as well as a *personal research project* involving real time animation in a competitive setting titled Space Travel. This was for an almost forgotten PDP-7 "with good graphics capability." Thompson and Dennis Ritchie implemented the predecessor to UNIX in assembly language on the PDP-7 to enable Thompson to get the Space Travel video game working (One can only guess how history might have been changed had AT&T understood the worth of video games as well as it understood the worth of UNIX.) UNIX was then moved to the DEC PDP-11 and subsequently rewritten in the new C language developed by Ritchie. The rewrite was completed in 1973. The use of a high level language to implement an operating system was unique for the day and would later be one of UNIX's important features.

Because of Federal antitrust rulings, AT&T could not sell UNIX, but it could give it away. In a brilliant master stroke (or a lucky move) AT&T started giving UNIX to universities. The smallness of the system and the fact it was written in a high level language made it attractive for teaching purposes. The fact that hundreds of students were adding to the system and growing to become disciples didn't hurt its current and future popularity.

By 1977, UNIX was being used in over 500 sites. Also in that year, UNIX was ported to the first non-DEC machine, the Interdata 8/32. From that year until 1982, several versions of UNIX were available within the Bell System. These were ultimately coalesced into one system called System III which was offered commercially in 1982. Several new features were added during 1982 and in January of 1983 AT&T offered official support of System V. (The missing System IV was never commercially released and enjoyed fleeting popularity within the Bell System during 1982).

The boys at Berkeley, being an unruly lot, did their own UNIX developing and came up with several additional versions. These were BSD 4.1, BSD 4.2 with the most current being BSD 4.3.

This history tells a great deal about the operating system. First, its design was largely motivated by intellectual curiosity and not market pressures. Unlike proprietary operating systems, UNIX did not have to compete with other venders nor support upgrades from previous systems. Second, even though the basic design for UNIX came from only a few minds, thus insuring conceptual integrity, a great many people have had a go at the system since its beginnings. As a result, UNIX is the product of evolution as opposed to design.

When comparing UNIX to other operating systems, this shows. For example, the UNIX human interface has little uniformity to its syntax and is filled with commands which represent the various implementers pet names or individual senses of humor. Contrast this with the VAX/VMS command language which was designed as a unit, with similar formats and relevant English names for each command. It's argued that these differences are only a problem for inexperienced users, but they are still a consideration. The lack of consistent design is also apparent in the system services. There are some calls which appear to have duplication of function as well as a lack of consistency in doing things. Error condition indication is an example. Most calls return errors in the same way, but there are a few which don't. Not that this is a serious problem, but it does tend to foster misunderstandings among new users.

UNIX's family history also points out another important problem. Everyone who's ever taken an operating system course knows something about UNIX and there are a great many people who know an awful lot. The result of this is that security on the UNIX system is difficult to control. Contrast this with something like the MPE/V system. Either by plan or carelessness, it's downright difficult to get a detailed picture of what's going on inside the 3000. This makes security a lot easier because there is already a confusion factor about what's happening. With UNIX's public privates, this is much harder. This problem is compounded by the fact that UNIX used to come with the sources right out there for all to see; just pleading to be modified, studied and tampered with.

The security problems with UNIX are horrendous even in UNIX heartland. At Bell Labs in New Jersey there has been a mini-crisis with unauthorized access to internal UNIX systems. Some, in darkened rooms with the backs to bright windows, have even admitted that there is fear for the master sources kept in Short Hills. It would be possible, some theorize, for hackers to subtly modify the system which is shipped thus enabling them to get in any derived UNIX system. This might go unnoticed for long periods with predictably disastrous results. Bell is taking herculean steps to correct the problem, but the fundamental reason for security difficulties remains the basic philosophy of UNIX and its general exposure compounded by UNIX's roots being in educational institutions. Further standardization activities by AT&T and the IEEE may actually worsen the problem.

So the conclusion is that unlike most proprietary operating systems, UNIX has grown by evolution often at the hands of research types. The result is that it lacks an overall consistency. In addition, the fact that UNIX's sources are generally available and well understood by many smart people introduces security problems which are unique to UNIX.

Something has to be missing

When comparing UNIX with other operating systems one thing should quickly strike you as odd. The basic UNIX kernel was implemented over several years by essentially two people. Take this in contrast to something like System/360 from IBM. This (monster) operating system took man centuries to implement as compared to a man decade for UNIX. We can conclude from this that either the boys at Bell are pretty smart when compared with the people from IBM, or that something is missing from UNIX which others thought important to include in System/360. Throughout the UNIX literature the *small is beautiful* theme reoccurs. Admittedly one of the basic system design goals was to provide a minimum amount of kernel function. Non-kernel user programs would be left with the job of providing the real sophistication.

This philosophy has several results. First, it is not true that small and easy to understand necessarily lead to optimum machine usage. Much of the complexity of other operating systems stems from the fact that they are giving the user a wealth of choices which offer various degrees of optimality for different programming situations. A good example of this is the difference between the process scheduling algorithms found in VAX/VMS and UNIX.

VMS offers the user with a complex set of scheduling techniques clearly breaking the UNIX small is beautiful rule. In fact, the VMS scheduler offers two distinct scheduling principles. UNIX, on the other hand, only offers one. It's much easier to use and, for the most part, transparent to the programmer. The problem is that a real time application has different scheduling needs than an interactive application. VMS offers solutions to both problems at the same time, UNIX offers only the interactive solution.

A second result of the "UNIX philosophy" is that UNIX users tend to view rolling your own as a normal way of dealing with operating system deficiencies. The file system doesn't perform the way you want? Just write your own. Does memory management miss the mark on important features? Just write your own. You either view this aspect of UNIX as great (because you can modify UNIX so easily) or terrible (because you have to modify UNIX so often). I know of few people who get in there and tinker with MPE, VMS, RTE or VM and even fewer who expect that they will have to.

This ability to tinker, often coupled with a real requirement to introduce new operating system features, tends to create a new class of programmer in your organization -- the UNIX OS Guru. Unfortunately, this is just the sort of thing that you were hoping to avoid by going to UNIX. Now, all of a sudden, you have pockets of unique specialized knowledge in your organization. No one really knows what changes were made. You are held captive by subordinates who have the keys to the kingdom. At least with a proprietary operating system you are being held captive by another (large) company with, hopefully, well understood motivations.

The list of features missing from UNIX is often long and may be important depending on your individual application. Most UNIX advocates dismiss "missing features" by telling you that there is some Berkeley version or third party package available which solves precisely those problems. This answer, to me anyway, is admission to a larger problem and that is the rapid proliferation of nonstandard UNIX systems. Among the missing features are the following.

A classic file system.

The UNIX file system is unique among operating systems. It reflects UNIX's original text processing application in the Bell Labs patent department. A file in UNIX is a string of characters with no record boundaries. A file consists of a number of blocks which are hung off of a master block called an inode. The inode block contains pointers to the other blocks. To extend the potential size of files, UNIX provides pointers from the inode block to subordinate blocks which, in turn, contain pointers to the actual data blocks. This results in the odd situation where the first part of a file is slightly faster to access than the last part of the file (last part requires occasional double reads).

Because UNIX pre-reads blocks into a buffer cache, reading a file sequentially results in fast access. As you are processing one block, the system is getting another for you. There are some problems, however.

Although a conceptually simple file system, UNIX gets into some trouble because of it. In fact, the file system is usually the first thing serious UNIX users begin to modify (just to make it run better). The usual file system problems include:

Too simple a file model. There are times when the classic record oriented file system is just what the doctor ordered. UNIX files have interesting properties, but they can miss the boat when it comes to large efficient data base applications. This criticism stems from the fact that the record model gives the user control over some of the physical attributes of a file which are important to performance tuning. In addition, many operating systems, give important system support to data base functions rather than force them to completely reside in applications programs as does UNIX.

Scattered blocks. UNIX provides little control over the allocation of files on the physical disk. The result is that data blocks are scattered all over the place. This causes multiple seeks when accessing the file. There is no way to cluster the file's data blocks to minimize the size of the seeks either. Contrast this with MPE, RTE, VMS, and other classic file systems. Files, or at least file extents, can be allocated in physically adjacent regions thus minimizing disk head movement. In fact, some operating systems even give you control over file placement with respect to physical disk cylinders so that head switching may be used instead of head movement. UNIX offers no such help. Some program which may be run after the fact allow you to reorganize the disk to provide more optimum file block locations.

No pre-allocation. UNIX gives you disk space as you use it. This means that when writing to a new file, disk allocation overhead will be intermixed with write overhead. There is no way to pre-allocate a file so that the allocation overhead is concentrated in one spot in your application. This problem is especially vexing to real time applications which may need to write data quickly and can't stand the allocation overhead at any particular moment. Because UNIX allows files to have holes in them, you can't simply position to the last byte of the file and write it causing the operating system to allocate all the other bytes. Doing so would make a file with only the last block in it and no others.

Other operating systems provide the capabilities to pre-allocate either all or some of the file when it is created. MPE's preallocation scheme provides a compromise solution which enables the user to anticipate the file's ultimate size and then determine how much should actually be allocated at file creation time.

Small data block size. The block size used for UNIX is either 512 or 1024 bytes. This is fine for interactive applications and text storage, but for large data transfers, it limits data throughput. Other operating systems don't have this problem, because they offer the user the concept of the record as a user controlled block size. User's may define big records which more efficiently accommodate larger individual /O requests.

Asynchronous disk writes. UNIX always performs disk writes to a disk cache. Because of this, the application program is never really sure that the /O has been performed to the physical disk. In transaction applications which wish to implement bullet proof error recover techniques, this is a difficult situation. Transactions are typically considered complete only when everyone is safe back home on the disk. UNIX prevents this information from getting to applications. Again, other operating systems, have extra features which either disable cacheing or allow the requesting program to specify synchronous writes.

Super block. Important parts of the file system are in memory in a UNIX computer. If the system crashes, the disk and the memory are out of synchronization. This causes damage to the file system. One especially vulnerable structure is the super block. The super block is an in-memory data structure which maintains information about the disk space managed by the system. There is only one super block for each file system (a disk may, however, be partitioned into multiple file systems which are then linked together). This is a dangerous design for systems which must have highly reliable data bases.

Sophisticated process scheduling

UNIX offers only one flavor of process scheduling. This technique was designed to give good performance to terminal jobs in an interactive environment (program development, text processing). Processes are given a time quanta and a priority. Processes are preempted on the basis of priority, but the priority level is dynamic and is automatically readjusted by the operating system when a process is suspended. High priority is given to new processes to insure quick initial response. The priority decays as the process runs (This type of process scheduling is difficult to simulate. The result is that UNIX is hard to model for system response time studies.)

Contrast this scheme with the priority scheme found in the HP RTE operating system. Typical of real time systems, the RTE provides simple priority preemption without priority adjustment. This gives the system the ability to insure processes fixed amounts of non-preempted CPU time when responding to events.

The VAX/VMS operating system, totally ignoring the "small is beautiful" credo, offers both types of scheduling. Processes with certain priority levels are scheduled with time share techniques while 16 priority levels are reserved to implement the real time technique.

Sophisticated Kernel

The kernel organization of UNIX is quite simple, but again, the missing parts are considered vital in some circles. For example, the UNIX kernel is non-preemptive (sounds pretty complex, huh?). This means that once UNIX enters its kernel, it won't interrupt out to begin another kernel task (other than /O interrupt processing) until the current task is completed or until the process which requested the operation is blocked. In English, what this means is that when a process requests a kernel service, the process begins execution in kernel mode. From that point on, the process cannot be preempted by another process in the system until the kernel mode request is either completed or until the kernel decides that the requesting process must wait for some other event to complete.

The net result of this is that UNIX processes can disregard other processes for as long as one second. Again, not something real time enthusiasts rave about in a positive way.

The HP/RTE system has the same problem, but it was written to try and be quick about getting out of the kernel. In addition, RTE has been modified to include a priority interrupt which is a way of actually interrupting the executing RTE kernel to address a more pressing need elsewhere.

VAX/VMS has an even more elaborate solution. It has an asynchronous kernel which is reentrant. It may interrupt itself and go off to process more important kernel functions while in the middle of processing a less important kernel function. This results in a larger more complex kernel, but provides far superior interrupt response and system throughput.

The realization that UNIX is a synchronous kernel, led HP to modify System V when producing HP/UX. They changed the standard kernel code to incorporate interrupt points. These are areas where the kernel may be interrupted by other kernel processes. This does not result in a truly asynchronous kernel, but it is much better than standard UNIX's strictly synchronous design.

Like many other UNIX vendors, HP has taken a hard look at the short comings of UNIX such as those mentioned above. In the case of HP, the HP/UX offering addresses each of the UNIX problems in some way. Although HP is lobbying to have their solutions accepted as "standard" the jury is still out. Every vendor who as tuned UNIX to address its problems wants their solution accepted as standard. It would appear that true standards will be long in coming. It would not be surprising to see that there will be several different standards evolving. Certainly not quite what user's have in mind when they think of portability.

Another final comment is about the richness of the kernel. Other operating systems tend to offer more in terms of kernel functionality than does UNIX. This is often explained by saying that the UNIX kernel offers the base upon which users write applications to perform the real work. This is actually what has happened. The UNIX kernel is surrounded by many man years of excellent software which is available to the UNIX terminal user. The problem is that transaction oriented systems tend to need the functionality at the process level. To have application programs providing features which might more appropriately belong in the kernel, precludes their use from other user processes. Once again, you are forced to roll your own.

And in conclusion...

Upon rereading what I have written detect a distinctly negative feeling. I don't think that this is intentional, because as a programmer, I use UNIX and like it. Perhaps I have presented some of the more negative aspects of UNIX because it is so easy to hear only good things about the system. From what you have read, I hope that you make the following conclusions.

UNIX is definitely a serious force in the market place. There are well over 200,000 UNIX installations. From that consideration alone UNIX must be doing something right.

UNIX does represent our best shot at a vendor independent operating system, but by no stretch of the imagination should you think that UNIX is standard. When you hear someone remark that they are using UNIX, you must ask a number of questions to really understand what they are saying. In addition, standardization itself, which on the surface appears to be so attractive, has its own innate problems; not the least of which is security.

UNIX evolved as an operating system instead of being designed from the start as an integrated whole. This is perhaps true of any mature operating system, but it is especially true of UNIX. As a result, there are areas of inconsistency which can be confusing.

Finally, UNIX is small. This is often confused with charming. UNIX's smallness is as much a result of missing functionality as it is the result of a good sparse design. To some, the omitted parts will not be missed, but to others, UNIX's lack of features will forever label it as a toy system still smacking of its research upbringing.

In all cases, though, it borders on silly to make blanket statements about UNIX. As with art, UNIX can only be evaluated in the context of its intended use. The potential user will gather the facts, weigh them against the application and only then conclude whether or not UNIX is the right or wrong solution.

TITLE: What Do You Mean, "The Job Blew Up"?

AUTHOR: Michael Madigan

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 0152

Conversations over the stable door

Anthony Furnivall

Buffalo News Inc,
A Division of Berkshire Hathaway
Buffalo NY 14240

Conversations over the stable door
0153 - 1

This paper is not meant to be an exhaustive discussion of the security implications of the HP3000. Even if I were qualified to take on such a subject, the time needed to do it justice would far exceed the time available. Instead, what I will attempt to do is to present my own thoughts on the subject, and then perhaps we can explore together the implications of some of these thoughts.

There are two sides to the security coin, just as there are two sides to most important issues. On the one hand we have the needs of the user to be protected from long, involved sequences of commands in order to complete a task; to be protected, in effect, from the possibility of an error, a potential security violation. On the other hand, we have the need of an auditor to protect the data from the same sort of errors, or from the possibility of intentional violations. These needs are largely incompatible, and it falls to the system manager to attempt to keep everyone happy.

We have a classic good guy/bad guy conflict. We can find a similar conflict anywhere we look - the right to privacy and the public's need-to-know, or the rights of an employee and the needs of an employer. There are other aspects of this conflict that readily come to mind, but let us first examine the basic perspectives of the computer user, and the most typical conflicting perspective, that of the DP auditor. (Fig. 1) The user has explicit needs NOT to be bothered with extensive verification procedures, and NOT to be forever passing security checks. However, the user also has definite needs to be asked to verify intentions at critical points, to be saved from the possible consequences of his or her actions. This model of the user as "good guy" presupposes basically good intentions, a small amount of curiosity, and a fairly good grasp of what is possible, but an overwhelming need to get the job done as fast and as soon as possible.

Compare this with the need of the DP auditor, who requires the system to be able to go back and recover every key stroke of every session, to be able to identify who did what and with what and to what, and to be able to report the slightest apparent malevolent intent as soon as it becomes apparent. This need is in self-evident conflict with the need of the user, and would indeed represent an intolerable burden if it were to be imposed only by manual means.

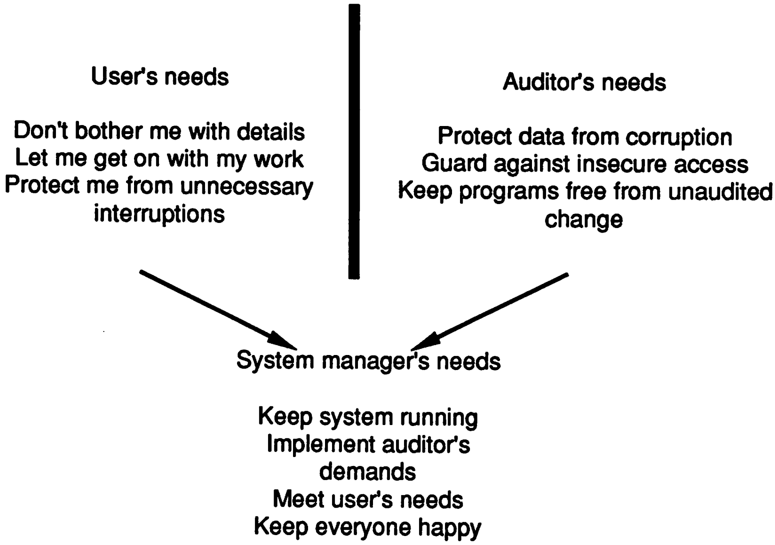

User's needs
Don't bother me with details
Let me get on with my work
Protect me from unnecessary interruptions
Auditor's needs
Protect data from corruption
Guard against insecure access
Keep programs free from unaudited change
System manager's needs
Keep system running
Implement auditor's demands
Meet user's needs
Keep everyone happy

Fig. 1 - The two sides of the security issue

By contrast, the system manager or program designer has already got enough on his or her plate before worrying about the apparently unreasonable and conflicting demands of the 'User' and 'Auditor' roles. It is only too easy to set up the basic minimum of security and let things go at that. This has the added benefit of leaving time to track down disappearing disc space, unexplained program aborts, degraded system performance and other non-trivial problems.

Even though we have identified three basic role-models for those who worry about computer security, they are as I have said, reducible to two. The conflicting demands must be mediated by whatever means we choose to use when automating the security of our systems. In order to begin to do this we need to be aware of the 3 essential components of a security violation. These three components are Access, Awareness and Action. It is in the way that these three components combine that we have the different needs of a user and an auditor. Let us examine, for a moment, the validity of this proposition.

○ *Access*

○ *Awareness*

○ *Action*

Fig. 2 - The three components of a security violation

In any computer system, there is a set of functions that are available. Each one of these functions has been designed for a specific purpose, and has been included for a good reason. Examples of such functions may range from reducing the balance of a loan by the amount of a payment, all the way down to providing a trace of the result of an IMAGE procedure call. The point is that every function has a purpose, an intended audience of users, and a set of precedents that determine the appropriateness of its use.

Now let us see how this set of functions fits in to our model of Access, Awareness and Action. To begin with, since the functions have been implemented by the designer, they must be accessible to at least some users. As an example of this, consider a standard that we have in place at the Buffalo News. (Fig. 3)

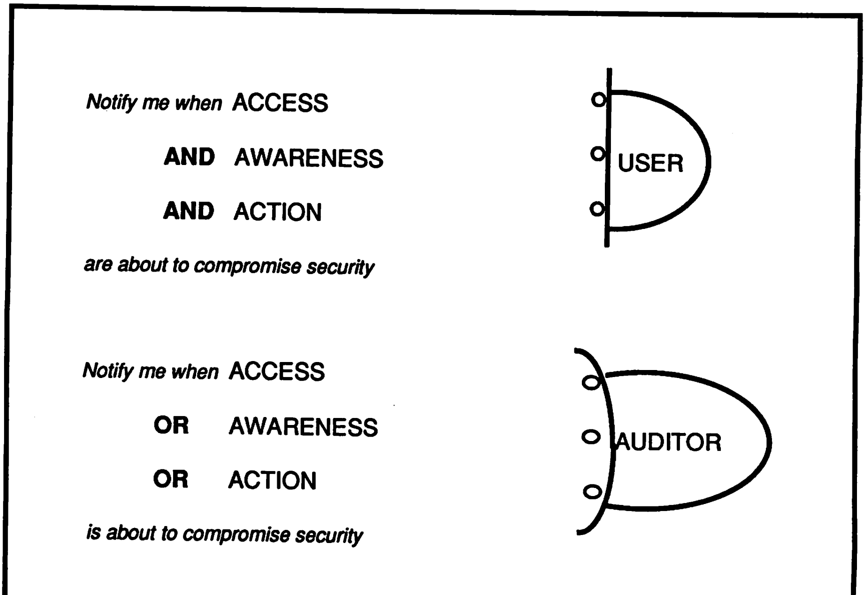
```
RUN CP525.PROG.CIRC;LIB=G
CP525    - (A.01.013) Carrier DM & Intro labels
CP525    - Tue, Apr 26, 1988,  4:55 AM


RUN CP525.PROG.CIRC;LIB=G;PARM=64
CP525    - (A.01.013) Carrier DM & Intro labels
CP525    - Tue, Apr 26, 1988,  4:55 AM
CP525    - CP525.PROG.CIRC          PIN 155 Parm=%000100
CP525    - TONY,MGR.CIRC,PRUN               Mode=%000007
CP525    - Flags: Test=N  Trace=Y  S220     Ldev 22
CP525    - Trace-file=CP525TR
```

Fig. 3 - Activating a feature of which the user may be aware

For programs that we design and code ourselves, running the program with ;PARM=64 will generally produce a trace of the program's action. This allows us to repeat a run which produced an error, and get a detailed internal listing of what went on. It is a useful diagnostic function which is intended for use by systems analysts in the event of problems. It is NOT intended for normal use because of the possible performance impact, and the possibility that a very large report would be generated, thus wasting both CPU cycles and paper - both of which are scarce commodities. ACCESS to this function is granted to all users who have access to MPE, and some of them are AWARE of this functionality. However, unless they choose to RUN the program in this way, they will not cause a violation of the intent of the function.

In much the same way, let us consider for a moment the payroll department of a company. Here is a group of people who are provided explicit ACCESS to some of the most sensitive information in the company, and who are very AWARE of the sensitivity of the data. Again, unless we have the explicit ACTION of passing this information along to unauthorized persons, we do not have a security violation.

Here is where the different needs of the user and the auditor come into sharp distinction.(Fig. 4) The user expects warning and notification ONLY at the point where a security violation is about to happen; the auditor wants notification at the earliest possible moment.



Notify me when ACCESS

AND AWARENESS

AND ACTION

are about to compromise security

Notify me when ACCESS

OR AWARENESS

OR ACTION

is about to compromise security

Fig. 4 - Different noptification needs of user & auditor

For the user, all of the components of the access/awareness/action model have to be active in a 'suspicious' context. For the auditor, the moment any 'suspicious' act occurs notification is necessary. Along these lines, we can see that any of the three elements of the model can be the source of a breach of security, and furthermore, that the determination of a violation is not easily made. For this reason, most computerized data security mechanisms concentrate on the ACCESS phase of the model, because this the easiest one to secure.

The last model that we need to consider in a preliminary examination of the issues is the model of the components of a secure system. We need to consider this model in the light of the others - that is to say, in the light of the different needs of a user and the system, and bearing in mind the three areas that need to be monitored.

WHO?	*HOW?*
The actual user - a real person OUTSIDE the system	The logical person - an entity in the accounting structure MPE LOGON = USER.ACCOUNT
The terminal- a real place with a connection to the computer. Ldev number?	The program - a set of functions - some are OK some are NOT OK
WHERE?	*WHAT?*

Fig. 5 - Four components of the run-time environment

This model of computer security focuses on the four basic questions that reporters are trained to ask every time they begin to write a story -the Who, What, Where, How questions. As we shall see, in terms of computer security these same questions apply. Let us consider these four components one by one, and see how they contribute to a fuller understanding of our other models. I propose to deal with them in the following order: Who, How, Where, What.

The question of WHO is doing something on a computer system seems, at first sight, to be almost trivial. (Fig. 6) After all, we have to identify ourselves at sign-on time, and this determines who we are. Or does it? Using MPE as a model, we can see that USERs are related in a sub-ordinate way to ACCOUNTS, and thus are explicitly differentiated between accounts. MPE enforces this separation almost without exception (the :ALLOW command does in fact allow an orthogonal view of the USER/ACCOUNT relationship). Our need is for a relationship which is super-ordinate to the accounting structure; an ability to model a pervading user who can (with any luck) access ANY account. The accounting structure, then, does not offer any support for our concept of the pervading user. MPE does provide such support, however,in the optional session-name, or job-name component of the sign-on sequence. Since this is not related in any way to the accounting structure of the system, we are free to

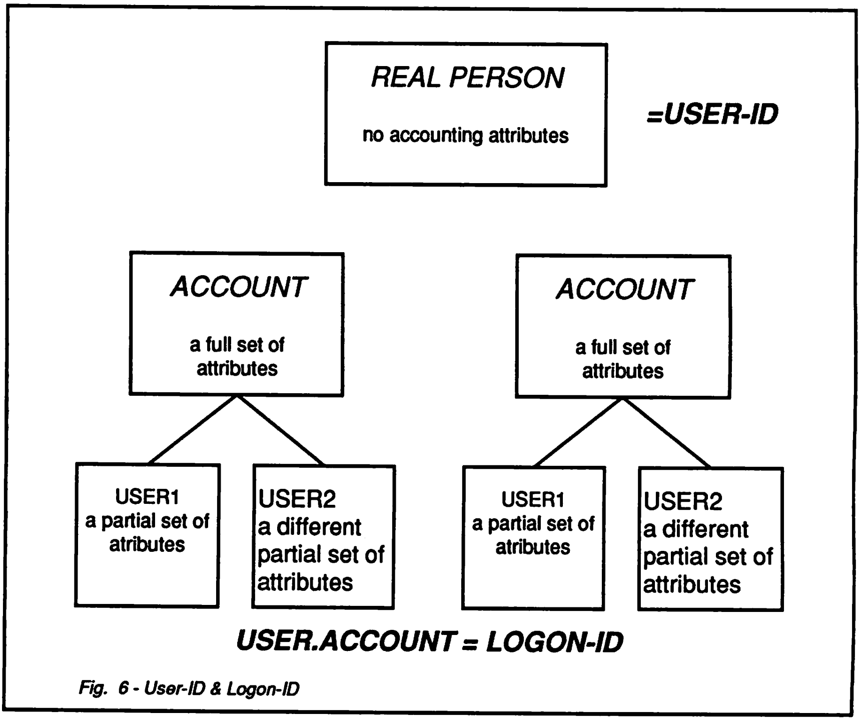

REAL PERSON
no accounting attributes
=USER-ID
ACCOUNT
a full set of attributes
ACCOUNT
a full set of attributes
USER1
a partial set of atributes
USER2
a different partial set of attributes
USER1
a partial set of atributes
USER2
a different partial set of attributes
USER.ACCOUNT = LOGON-ID

Fig. 6 - User-ID & Logon-ID

use it as a model of the real person who is using the system. This model I propose to call the USER-ID, to enable it to be easily identified later on.

The next question we need to ask is HOW is the user represented inside the computer system. This is where the computer's accounting structure becomes of primary importance. From the MPE perspective, when we log on to the 3000, we specify a combination of account and user which together constitute a LOGON-ID. This logon-id has certain attributes associated with it, and these attributes are used extensively by MPE to determine the permissibility of almost everything that the user wishes to do. Some linking of the logon-id with terminal attributes is done at logon time, but this is rather limited in nature. It is important to note that MPE allows a logon-id to have a subset of the attributes available to the account. For example if an account is granted SM capability, there is no need for all the users to have this capability. Depending on which LOGON-USER is in effect, the capabilities of the account will be restricted to the set that has been granted to the LOGON-USER.

Our examination of the four part model of computer security needs also to consider the WHERE question. What I mean by this is, "Where is the actual user presently located, and is it appropriate to allow this request to continue?" It is easy to imagine instances when this question needs to be asked - if the user is seated in the front lobby of the building, it might perhaps NOT be a good idea to allow the salary of the president to be displayed. Similarly, if a given function is requested by a terminal located in a user department that is more appropriately performed by the system manager's terminal, this might also be grounds for the system to consider the request a violation. Note that, in this regard the terminal has some of the attributes of both a user and a resource.

Unfortunately, the question of identifying the location of the user is a vexing one for two situations, and in these situations MPE is not a very helpful partner when it comes to tracking down potential security violations. (Fig. 7) These two areas are batch jobs and data-communications devices. For a batch job it would be desirable to know the source of the batch job - that is to say, who (in the model's sense of the word) streamed the job, and what file was used (including $STDIN). This monitoring would need to be extended down through however many levels of jobs are initiated by the original stream command. This is provided at the present time by various stream management packages, but, as far as I am aware, the data is NOT available programmatically to processes running under those jobs.

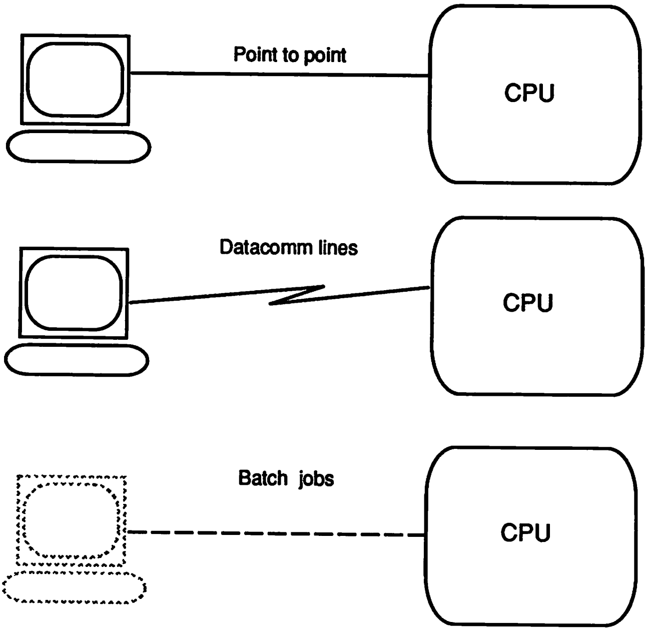
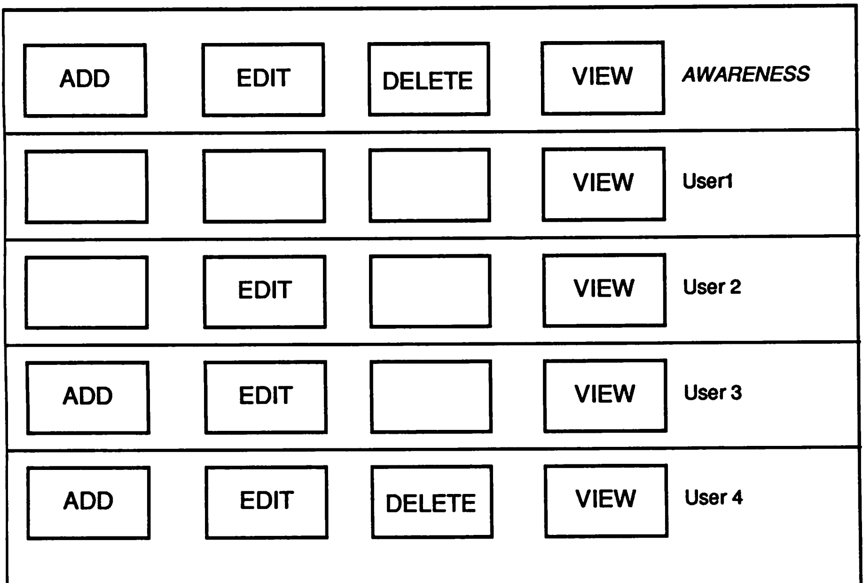


Fig. 7 - Terminal location, and the question of 'Where?'

A similar situation exists for those logical devices on an HP3000 machine which are connected via modems, data switches and other similar boxes where the end to end connection between the port of the computer, and the terminal is not fixed and unchanging. Such a situation obviously covers a large number - possibly even a majority - of cases. In these situations it is also impossible for a program to determine the true location of the user, and thus to include this in its own decision making. NS/3000, the software component of HP's local area networking product does in fact address this problem, and it is possible to work backwards through an NS link and discover where the user is located, although this may be a rather torturous process.

Considering the WHAT question is considerably easier - computers use programs to change the data, and the WHAT of our model is unequivocally related to the program that our user is running. While the model is easy to define in this instance, it has some disturbing implications.



ADD	EDIT	DELETE	VIEW	*AWARENESS*
			VIEW	User1
	EDIT		VIEW	User 2
ADD	EDIT		VIEW	User 3
ADD	EDIT	DELETE	VIEW	User 4

Fig. 8 - Programmatic functions

If our model is to be aware of and sensitive to the complete environment, that is to say the combination of USER-ID, LOGON-ID, TERMINAL-ID and PROGRAM-ID, this implies that programs may need to produce different results for different users! How many programs do you know that are capable of dynamically adjusting themselves to a different user? IMAGE does in fact include this capability, but the programs that use IMAGE procedures need to accomodate the varying results of IMAGE calls. It seems likely that most programs do NOT provide a dynamic functionality based on the actual user-id, but rely instead on some other means of preventing abuse.

As an example of what I mean in this regard, I would like to present a brief model of a typical data-maintenance program. (Fig. 8)This program uses function keys f1 thru f4 to request Add, Edit, Delete and View functions. From the point of view of consistency, all data-maintenance in a system will use this model. It seems reasonable then, for a program to disable these functions when they are not appropriate for use by a given user-id. The user may well know that f1 means Add, but the program must not only disable the 'awareness' of the user (by hiding the function key label) but must respond in an appropriate manner to both the user and the system manager. The user needs either a simple message that the function is unavailable, or preferably, a null response from the program. The system manager, or the auditor may need to know that an attempt was made to execute a function that would have resulted in a security violation.

```
FILE SENSITIV;DEV=PRIVATELP

FILE SENSITIV;DEV=PUBLICLP

FILE SENSITIV;DEV=DISC;SAVE

FILE SENSITIV;DEV=MODEM

FILE SENSITIV;DEV=IBMPC
```

Fig. 9 - Run time detection of report destination

Another example of the need for a program to respond to its run-time environment is found in the ability of a file equation to redirect the destination of a report. (Fig. 9) What may be intended to go to a private line-printer may end up being directed to a public line-printer, a modem or a PC! This is something that can only be detected at run-time, but at least it can bedetected.

MPE then, provides a reasonably good matching of the four phase model of computer security, and provided we can access the attributes that are provided, and define relationships between them, we ought to be able to make a good decision about the permissibility of any request by a user. One of the important benefits of the four component model is that once access to the computer system has been established, three of those components do not change. (Fig. 10) (For now I will exclude the problem of an unattended terminal).

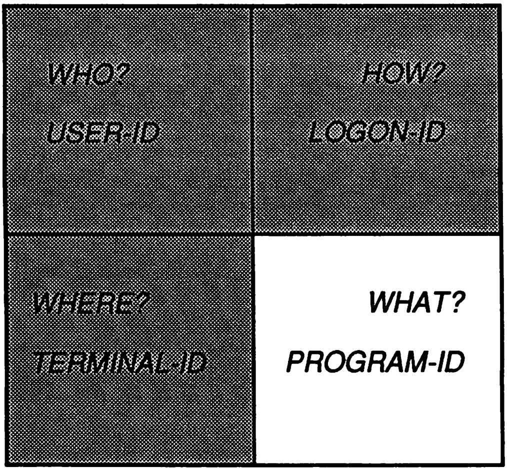


Fig. 10 - After logon, most of the components remain the same

This makes the task of security checking easier - since part of the checking is handled once at logon time, and can then be implied after that. We must remember though, that we have explicitly included an entity outside the knowledge of MPE, and the directory structures. For this reason, we are faced with the fact that the only way we can determine the legitimacy of access to the system is to allow access, and then deny it if necessary. Further, to provide true security, we need to find some means of determining whether any potential violation of access or awareness is contained in any given request for an action to be performed.

It is as we consider this need that the importance of all three models becomes apparent: The user has different needs from the Auditor (or the auditor's representative - the system manager); we need to monitor access, awareness and actions, and we need also to keep in mind the binding represented by the user-id, the logon-id, the terminal-id and the program-id. What would be reasonable would be to consider various strategies for keeping track of all these entities, and possibilities.

If we are to do anything to meet the needs of the system manager, it is obvious that we must monitor every action. Equally obviously if we are to meet the needs of the user, we must not make such monitoring an obstacle to getting useful and productive work accomplished. Consider the common practice of providing employees with an identification badge. The purpose for this is to allow some mechanism (generally a security guard, or a card-key door) to identify employees, and to grant them access. Typically, such programs start out with a lot of attention, and the security guard is very active checking every employee for their badge. This is generally viewed as a hassle and as an obstacle to productivity! In addition, after a while it is possible to notice that the guards adopt a different strategy towards checking employees - they get to know who is who, and the frequency of checks becomes less. What we need to find is a way for the intrusion to be minimized, but for the vigilance to be maintained.

MPE provides a very good example of incorporating security consciousness into its activities. In addition, it does a fair job of meeting the conflicting needs of the user and the auditor. I shall show how this occurs, and then go on to discuss the strategy used to grant approval.

```
USER'S TERMINAL

ENTER LOGON PASSWORD (MGR):

ENTER LOGON PASSWORD (MGR):

ENTER LOGON PASSWORD (MGR):

INCORRECT PASSWORD (CIERR 1441)


SYSTEM CONSOLE

13:12/#S545/229/INVALID PASS FOR "TONY,MGR.CIRC,TRUN" ON
LDEV "65"

13:12/#S545/229/INVALID PASS FOR "TONY,MGR.CIRC,TRUN" ON
LDEV "65"

13:12/#S545/229/INVALID PASS FOR "TONY,MGR.CIRC,TRUN" ON
LDEV "65"
```

Fig. 11 - MPE accomodates both reporting needs

When a user fails to provide the correct password at logon time, MPE notifies the console immediately, reporting the complete logon string, and the logical device. (Fig. 11) The user receives another prompt, with no indication that the first attempt was inaccurate. Only after 3 failed attempts does MPE indicate that the user has supplied the wrong value. Notice how our two conflicting needs have been met. The system has supplied notice to the system manager at the earliest possible moment, and the user is not obstructed until after three attempts. An area where MPE does NOT do as good a job of taking care of the system manager's needs is in the area of reporting file security violations. These are reported back to the user, and can be handled as appropriate or necessary. However MPE provides no notice to the console of such a security violation. New versions of MPE will permit the logging of file opens, and this will help in some way to rectify the situation.

Permanent attributes - *an intrinsic part of the entity*
Eg SM, AL, PM, etc

Dynamic attributes - *granted at logon time, or process start*
Eg AC, GU, etc

Fig. 12 - Matching attributes

We have seen that MPE is evidently continually checking security parameters as it does its work. Two distinct strategies can be defined in the approach to security. The first is the strategy of matching attributes. (Fig. 12) These attributes are of two varieties - permanently assigned, and dynamically assigned. Consider the attribute AL, for account librarian. This attribute is available to the logon-id (since it is available at both the account and user level), and is always present whenever someone is logged on in this way. This attribute is used to determine the legitimacy of a file access.Or take the attribute OP. This permanent attribute is used to determine the legitimacy of (among other things) a STORE command.

The second sort of attribute is the attribute which exists ONLY for the duration of the current session or job. Such attributes include the Group User (GU) attribute, and the Account User (AC) attribute. These attributes are also useful in evaluating security, especially since they are a truer reflection of the present environment.

Another aspect of attribute matching is that attributes may be either required or permitted. (Fig. 13a) A required attribute is an attribute that must be present in order for a request to succeed. MPE uses its attributes in this way. For example, if I want to issue a NEWACCT command, I must have the SM attribute active in my logon-id. Similarly, if I wish to use data communications devices, I must have the CS attribute active. It is equally possible to disallow attributes. While MPE does not define any such attributes, they are easy to think of. Multiple logons would be a good candidate. (Fig. 13b).

Required attributes - *attribute must be present to gain access*
Permitted attributes - *attribute must be permitted if access is access is sought with the attribute active*

Fig, 13a

Multiple logons as Required attribute	*I must have this attribute granted to me in order to gain access*
Multiple logons as Permitted attribute	*If I am logged on more than once (i.e. the attribute is active), then the attribute must be permitted in order for me to gain access*
Net effect	*I may only gain access if:* *a) I MAY log on more than once* *and* *b) This is my current FIRST logon*

fig 13b

Fig. 13 - Various aspects of attributes

In this example, access to a logon-id would be denied if the access would result in multiple logons for the user-id. Note that an attribute such as this can be used in both ways. For example, it may make sense to limit access to a logon-id to those people who have the required attribute of multiple logons. It is even possible to model some pretty exotic situations using something as simple as required and permitted attributes.

Consider for example the possibility that access to a logon-id is available only to those who have the required attribute of multiple logons (i.e. they CAN log on more than once), but who are NOT currently logged on more than once (the same attribute is not a permitted attribute). We have effectively said that access to this logon-id is limited to the FIRST logon by a given user-id. Attribute matching in this way can provide a very strong degree of protection in ways that would be otherwise difficult to describe. Attribute matching, according to changeable specifications, is one of the two strategies that MPE seems to employ to manage the security function.

The second one is the concept of assigned rights. This rather awkward term is what I use to describe the ability of a file's creator to change the name. In effect, the right to do this is assigned BY the file TO the logon-id. This strategy is a very useful one, because it provides a 'hot-line' to a particular functionality.

Specific right	SPECIFIC ENTITY
Rename a file	CREATOR OF FILE
Use system console	LOGON AS OPERATOR.SYS

May be used to extend or limit access

Fig. 14 - Assigned rights, another strategy

Assigning a right is dependent on being able to define two things - the right that is being assigned, and the entity to which it is being assigned. (Fig. 14) It is easy to consider, for example, the possibility of extending MPE security to assign the right to use the system console to a particular logon-id (OPERATOR.SYS) for example. By assigning a right in this way we create a fixed set of rights that can be scanned for validity. If a right does not exist in the set then any request to exercise that right will fail. Another example of an assigned right is the way in which programs such as DBUTIL will operate only on databases in your logon group and account. Here what has happened is that the right to reference files in other groups has NOT been assigned. It is therefore missing from the set of assigned rights and, as a result, the request will fail.

It is interesting to speculate on ways in which MPE could be enhanced by using this concept. Perhaps we could assign a terminal to a specific logon-id. In this way we could guarantee that only PERATOR.SYS could log on at the system console, or perhaps we could limit use of the dial-in port to MGR.TELESUP (or possibly NOT MGR.TELESUP!). In addition to allowing specific rights, the concept can also be used to limit functionality. In the example we used above, it might be that the only device that OPERATOR.SYS can use is the console.

A third strategy that is not used for security checking in MPE (so far as I know), is the concept of security level. This is the same concept as priority level, except that what is being ordered is not the sequence of events, but their permissability. Let us see how such a concept works. (Fig. 15) We assign an arbitrary level of security to a logon-id, eg 4. Only users with a security level of 4 or above can access this logon-id. Or again, we assign a particular function within a program a security level of 3. When the program is run by a user with a security level of 2 or less, the function is disabled, and when the program is run by a user with a security level of 3 or more, the function is re-enabled. Several benefits accrue from this strategy. To begin with, it is a rule rather than a reality. We can compare the two security levels involved and gather our result based on a rule, rather than on the result of finding this specific instance specified. Compare this with the concept of assigned rights, where every possible instance must be accomodated. Using security levels, new entities can be added to the environment without modifying the set of assigned rights. Furthermore, if we want to restrict access to a logon-id, we merely raise its security level.

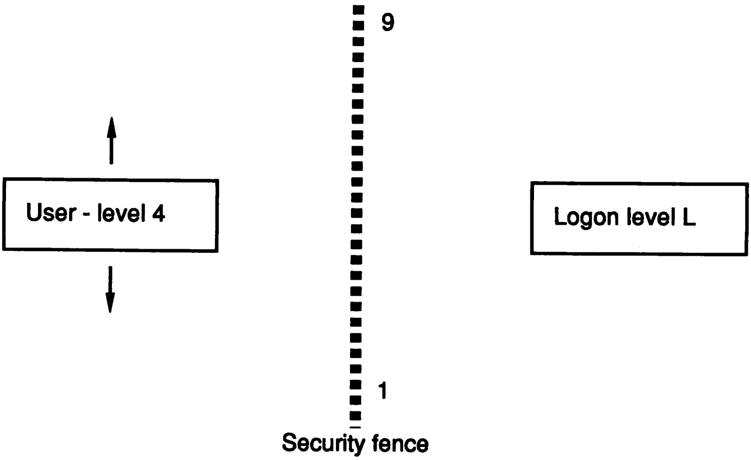


Fig 15. - Security level, a 3rd strategy

To summarize, then, we have three possible strategies for determining the approval of a request. They are: the strategy of matching attributes, either static, permanent attributes or dynamic attributes; the strategy of assigned rights and the strategy of security level. By combining these in various ways it is possible to achieve a very flexible degree of control over security requests. Even so, we are only handling explicit requests. Another large problem for the security function is the problem of nothing happening.

When a terminal is inactive for any substantial period of time, the binding of the user-id, the logon-id, the terminal-id and possibly the program-id becomes weaker and more tenuous. (Fig. 16)`There exists some threshold of inactivity beyond which we can not assert that the binding is valid. In some instances this may not be a problem, but in others it will be.

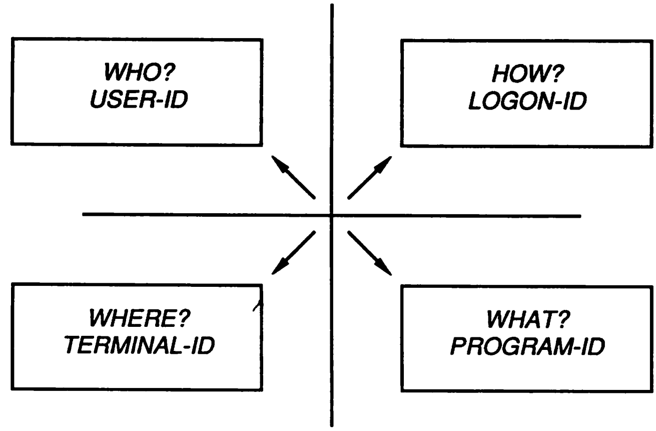


Fig. 16 - The problem of the inactive terminal

What we need to be able to do is to define the maximum period of time that any one of the four components can be inactive; and then, to define the action to be taken once this threshold is reached. If the period of time is of no consequence, then a value of 0 could easily indicate this. Possible actions to be taken would include re-verification of the user-id, or the terminal-id, forced termination of the program or forcing an end to the session. What is important to realise, I think, is that each of the components needs to be defined in this way. Only thus can we preserve the integrity of all four components.

The last area that I wish to address in this paper is the problem of data integrity. This subject warrants an entire paper on its own, but I wish to consider it from the point of view of ensuring that the program which is requesting permission to access and manipulate data is in fact the program that it purports to be. The idea of asking a person to prove who they are by supplying the answer to a question (such as a request for a passsword, or a personal knowledge question) is not new. However, it is not possible to ask for an animate response from an inanimate program.

It is easy to ask a system manager to ensure that only the right version of a program is allowed into production, but not so easy to make it work. We need to define a sufficient set of data to make it impossible (or at least sufficiently unlikely to be almost impossible) for a program to be a wolf in sheep's clothing. What constitutes such a set of data?

Program ID

Program file

Date & time of last modification

Protected group ID

Fig. 17 - Necessary data for monitoring program access

The first element we need is obviously the name of the program. (Fig. 17) This is not the same as the name of the program file, but the name by which the program is known to the programmers, designers and other people. In short we need the PROGRAM-ID. In addition, we need the name of the program file. This is a piece of data which can easily be extracted at run time. We also need to know the date and time of the last change to the program-file. Armed with this data, we can compare it with known values, and determine if the program that is requesting permission to proceed is in fact the program that is allowed to proceed. In order to allow testing, the checking is only done when the logon-id indicates that a specific protected group is being referenced, either directly, or by file equations.

The other aspect of a program which bears scrutiny is the frequency with which it is run. If a program is run more frequently than expected, it may represent a possible security risk. At the very least it is worth bringing to the attention of the system manager. What we need to specify in this instance is a range of time that we can reasonably expect between activations of the program. Naturally, for programs where this is of no concern, nothing needs to be done.

At the start of this paper I said that I would follow the implications`of my own thoughts on some security related issues. Where I think we have arrived is the description of various strategies for implementing the security function in the programs that run in your own shops. Some of these may be programs that are written in your own shops, and over these you can exercise as much control as you wish. Yet other programs may be aquired from 3rd party software vendors. Over these you can obviously exercise less control.

There has been much discussion recently about the importance of security, and the importance of being concerned about it. The place for such concerns is not only in the operating system software, but also in every program that runs on a system. Ultimately, the value of security consciousness is defined by the possibility of loss and the cost of loss. This is an equation which must be solved anew for every site. I hope that the ideas presented here will stimulate your own thinking, and lead you into defining your own solution to that equation.

TITLE: A New Model for Report Management and Distribution

AUTHOR: Michael A. Casteel

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 0154

New Paradigms For Automating Batch Job Processing

Michael A. Casteel
Unison Software
Mtn. View, CA 94043

Since the introduction of the Series II in 1976, the HP 3000 has steadily gained installed base and stature as a business computer. It is no longer just a minicomputer—the Series 70 delivers mini-mainframe power, and HP's new Precision Architecture offers the potential for full-scale mainframe performance in a supercharged HP 3000 chassis. More and more HP 3000 users find themselves running a full-fledged data center, and their companies' demands for computing productivity translate into demands on the computer operations staff.

Like their counterparts elsewhere in the company, data center and operations managers are now looking to the computer to aid them in the performance of their work, and they're using a new breed of software designed for data center management. The operations staff, like new computer users in other departments, must learn and adjust to new ways of doing their jobs. This is never easy, but change is necessary in the pursuit of greater efficiency and productivity. This paper attempts to ease the way by presenting the principles involved in automating one of the most important operations tasks: control of batch job processing.

Although clearly not conceived as a batch processor in the traditional sense, the HP 3000 has had to assume a batch processing burden commensurate with its growth as a general business computer. A number of HP 3000 data centers report they are now processing more than 20,000 batch jobs per month, with the jobs subject to many interdependencies. How can this volume of batch processing be reliably accomplished with the MPE :STREAM facility?

The answer is, it can't. The basic MPE facility only allows the scheduling of jobs based on time dependencies, e.g., hold a job until 8:00 p.m. before running it. There is no built-in provision for MPE running a job every day, every month, or every quarter, nor a way to ensure that a sequence of jobs is run in a certain order. In order to handle all their batch processing, HP 3000 users have resorted to a combination of manual operation and software tools such as SLEEPER from the INTEREX Contributed Software Library. However, this combination is inadequate to handle the jobs on a single Series 70, where six, eight, or even more jobs must be kept running to effectively utilize the machine's capacity.

Over the past few years specialized software tools have become available, mostly from third-party software suppliers, which provide the automation necessary to support production batch processing. These tools put the computer to work for the data center operations staff, just as existing applications serve the end-user. This means the operations staff must adapt to a new working environment, as the end-users already have. This paper presents five major functions which are fundamental to batch job automation, regardless of the particular implementation. They are:

scheduling	- to determine the jobs to be run on a given day
sequencing	- to ensure the jobs run in the correct order
constraining	- to keep conflicting jobs from interfering with one another
synchronizing	- to coordinate with external events
recovering	- because something will still go wrong, at least occasionally.

Scheduling

The first step in production batch processing is scheduling; that is, deciding which jobs are to be run this day. This is also the first, and perhaps most difficult, task to automate. It often takes many months before all the production jobs in a large shop can be identified and cataloged into an automated scheduling system, a phenomenon which underscores the need for automation in such shops.

Of course, not all jobs are scheduled in advance. For example, a programmer's compile job runs when the programmer has completed work on a section of a program. In a production environment, however, hundreds of jobs are scheduled on some regular basis, e.g., every Friday, every work day, or every month-end.

One of the most challenging problems in scheduling regular production jobs is the definition of the *calendars* used to determine when they run. Some processing might be scheduled with regard to the ordinary calendar (e.g., every Friday), and some by an artificial calendar, such as month-end processing for a calendar where each 13-week quarter contains one 5-week and two 4-week months. A shop may require references to several calendars in order to schedule, say, Manufacturing, Financial, and Payroll month-end. The scheduling task is likely to be further complicated by holidays, which may cause processing to either be skipped or rescheduled before or after the holiday.

An automated scheduling system must therefore provide for several different calendars, customizable to the company's holidays and other special requirements. Setting up the calendars is a prerequisite to scheduling jobs properly. Before trying to automate batch job scheduling, it is important to identify the scheduling calendars used, and then study the calendar functions offered in the automated scheduler to obtain the best fit.

In order to handle irregular jobs, which don't fit any calendar, it helps to be able to specify a list of specific dates on which to run each job, rather than calendar intervals. This way, if it is at all possible to predict the schedule on which an irregular job is to be run, it can be documented in the scheduling system and processed automatically. Examples of such scheduling are jobs which run at the Full Moon (I don't know of any automated scheduler which includes the Lunar calendar!), or jobs which run two weeks before each Interex conference.

Of course, every shop is likely to have a substantial amount of *ad hoc* scheduling, where user departments submit jobs or job requests on some basis known only to themselves. To minimize overhead, it helps if the automated scheduler will allow users to submit their requests directly to the scheduling system rather than to the operations staff. Once the users have been trained, their special processing requests can be integrated into the production schedule automatically, while operations concentrates on monitoring and controlling the batch process.

Another benefit of automated scheduling is the ability to schedule jobs *reliably* over long intervals, particularly quarter- and year-end processing. Manual scheduling of infrequent jobs is complicated by lapses of memory and staff turnover, while the computer never forgets.

Sequencing

Once the scheduler has determined *which* jobs should be run, the next step is to define the correct processing sequence. Although some jobs are essentially independent of others, it is critical that most jobs be executed in the proper sequence with respect to other jobs. Consider a batch update and reporting application, in which the processing sequence is: First, a backup (in case something goes wrong); then, an update job posts the batch transactions to the database; finally, a number of report jobs analyze the updated database.

This case illustrates the need for proper sequencing. If the update does not follow the backup, either it will fail because the database is tied up by the backup, or the backup will be useless because it did not get a complete copy of the database before updating. Furthermore, if the reports do not follow the update, they will either fail or produce incorrect figures.

The essence of job sequencing is this: A job must follow another job if it uses the results of the first job's processing (e.g., reports follow the update), or if it modifies a file which is required by the first job (e.g., update follows the backup). If a job does not use another job's output or erase its input, then it usually needn't follow it; it may be that the two jobs must never run at the same time, but that is a *constraining* issue and is covered in another section.

From this it may seem that the simplest way to define the processing sequence would be to list jobs in the proper order. For example, we would simply list, the backup first, then the update, and then the reports. By following this list, the computer operator can correctly process the jobs, one at a time (assuming nothing goes wrong). Many shops use just this approach for existing manual or semi-automated operations, and MPE even supports it: Just set the job limit to 1 and stream the jobs in the correct order.

Of course, ideally we want the HP 3000 to handle more than one job at a time. After all, it is a multiprogramming computer system. If we have several independent applications, such as accounting and manufacturing, each can have its own list of jobs to be processed independently. Although MPE offers little assistance, a number of such lists could be maintained and executed, depending on the skill of the operator.

But the simple list scheme becomes complicated when we try to take advantage of multiprogramming within one application. In the backup/update/report example, it may be that some or all of the report jobs can be run at the same time. What is needed is a simple means of maintaining job sequence dependencies with the flexibility to permit multiprogramming whenever possible.

Automated job processing typically uses an approach with maximum flexibility: For each job you specify which other job(s) must come before it. If we use the word FOLLOWS to signify this relationship, then we can easily express job sequences as, "UPDATE FOLLOWS BACKUP," and "REPORT FOLLOWS UPDATE." In the example we have been discussing, there may be several reports, all of which "FOLLOW UPDATE."

Clearly, this approach makes it easy for the computer to achieve the highest possible degree of multiprogramming. At any given time, the machine can process all jobs not waiting to "FOLLOW" jobs not yet complete. This approach also accommodates complex interdependencies, in which a job is dependent on *more* than one other job. A report may combine input from two applications, for example, and it need only "FOLLOW" update jobs in each application. This kind of job would be impossible in the listkeeping scheme, but is simply and elegantly handled by stating the individual job dependencies.

As it can take considerable effort to identify all the production jobs to be cataloged into the scheduling system, it can be even harder to determine the correct sequencing rules. One of the disadvantages of the simple list so often used is it doesn't reveal precisely *why* jobs run in a particular order, and this can make it difficult to recognize multiprogramming opportunities. If the list does not distinguish whether the sequence is due to true dependencies or simply because some jobs cannot be run together, it can take a long time to find out the necessary facts. Once rules are determined and then documented in the automated system, job throughput can be optimized. The information can also be invaluable in application system maintenance.

It is usually not enough simply to cause Job B to FOLLOW Job A. Usually, we require that Job A complete *successfully* before we can permit Job B to run. In our example, the reports should not be run if the update job terminated abnormally, due to something like a work file reaching its capacity. Instead, dependent jobs should wait until the problem has been corrected and necessary processing completed. Ideally, the batch processing software should signal the problem to a designated person, and perhaps proceed with automatic recovery processing.

An important consideration in this regard is how the software can know whether a job has succeeded or failed. Without automation, someone usually has to look at the job's output ($STDLIST) or read a completion message on the console. A common criterion, quite compatible with automation, is to consider a job successful only if it reaches the ":EOJ" command at its end. Although this doesn't hold for all jobs, experience shows most jobs adhere to this rule. The rule also covers those occasions when the job itself doesn't fail, nor does it complete, such as a system failure while the job was in execution.

If "successful" job completion is signified by reaching some point in the job other than the end, it may be possible to modify the job by inserting a utility program at the point which will signal success to the batch controller. Or, the job could be modified to adhere to the ":EOJ" rule, and only reach its end on successful completion. The advantage of the latter approach is consistency with the majority of jobs, always an aid in maintenance.

Finally, it may be that successful completion of a job can only be determined by inspection of the results. Software products now on the market can automate this by scanning reports or listings for tell-tale messages, but in general it may still require a person to check the results. To derive maximum benefit from automation, such applications should be modified to remove the need for inspection and provide a more obvious signal of success or failure.

Constraining

Now that the jobs have been scheduled and sequenced, there may be further constraints which need to be applied. One such constraint is illustrated by the MPE job limit: Only so many jobs are allowed to execute at one time. Some automated job controllers refine this function, offering the ability to limit the number of jobs executing for a given application, or in a given account. This can be a handy resource allocation tool. At month-end, for example, you can allow accounting a half dozen jobs at a time, while limiting development to two, facilitating month-end closing.

There are often more specific constraints to be applied, such as making certain that particular jobs never run at the same time. Although this can be (and often is) handled by sequencing one job before another, it can be counterproductive when there is no real reason for the sequence. If one job is arbitrarily selected to go first, any event which delays that job will also delay the second. It is therefore important to distinguish such dependencies from ordinary sequencing.

Exclusive job constraints are typically due to a conflict over some resource in the system, such as a database, a file, or even contention for the CPU. It is helpful to identify these resources explicitly when establishing processing constraints. First, this documentation removes any mystery regarding why certain jobs can't be run together. Second, it permits more effective maintenance, such as when new jobs are developed which require the same resources. Finally, the systematic recognition of resource constraints can turn up new opportunities to increase productivity. For example, jobs which require the system's only tape drive can run together successfully, but only one will actually be executing at any time. By identifying such jobs to the job controller, they can be kept from occupying executing job slots while waiting for a resource to become available.

Some automated systems extend this resource-centered model by distinguishing between "shared" and "exclusive" use. For example, jobs which access a database would identify it as a needed resource. Read-only jobs such as reports can specify shared access, which allows several such jobs to run at the same time. If *exclusive* access is specified for update jobs, the system can not only avoid running two update jobs together, it will not run an update while other jobs are using the database. This concept supports the highest degree of multiprogramming while maintaining the constraints necessary for successful processing.

Shared resources can be further controlled by designating the available quantity of the resource. If two tape drives are available, two jobs can be run if they each request only one unit of the "tape drive" resource. This approach can be extended to bulk resources like "CPU time" and "Disc I/O," and used to create a dynamic "job limit" which takes into account the special needs of certain CPU- or I/O-intensive jobs.

Synchronizing

The third facet of job control is synchronization with external events. For example, a certain input/update job must await the arrival by courier of a magnetic tape each night. Since this tape is not visible in the computer, the controller software cannot release the job into execution automatically. Instead, it is usually the operator's job to inform the software when the tape becomes available.

Automated job controllers commonly provide for operator input via prompts or "run book" codes, which the software displays and the operator responds to when the external condition is satisfied. In addition to the availability of input data, such conditions may include the reservation of a peripheral device or the state of an application database (are all on-line users off the system?). This method is sometimes employed for job sequencing, when a pre-dependent job is processed on a remote computer not in the local network, or when the successful completion of a pre-dependent job must be certified by someone before processing can continue.

Sometimes external events are visible to the software and synchronization can be automated. It may be that the availability of input data, or the shutdown of on-line processing, can be determined by the appearance or availability of a file or database in the computer. A number of automated job controllers offer such "file dependencies".

And, it may be that external software is able to determine that a synchronizing event has occurred. In this case, it is useful for a program or job to be able to send a signal, akin to the operator's response, to trigger dependent processing.

Recovering

Finally, there comes a time to face the inevitable: A job fails and some recovery procedure must be performed. The job controller's first contribution may be to announce the job's failure in a timely manner. The sooner the problem is attended to, the better the chance of getting production back on track.

In most shops, the ultimate (sometimes only) recourse is to "call the programmer." Most job control packages will allow you to attach some descriptive information to each job and make it available on-line, such as the name and telephone number of the programmer. There are even software and hardware packages which can make the phone call automatically!

Although the variety of possible recovery actions is as varied as the possible causes of job failure, it may be possible to establish a highly automated recovery procedure. For example, failure of a particular job could trigger the job controller to launch a restore of the database and continue with other processing, bypassing the failed job until it can be attended to. Other automated actions could include rerunning the failed job, or simply bypassing it without recovery.

Conclusion

The operations staff will succeed in automating their batch job processing without severe difficulty by carefully attending to these few basic elements:

Scheduling – Identify all jobs which are regularly scheduled and determine the basis for the pattern in which they are scheduled. Obtain the calendars used and understand holiday scheduling.

Sequencing – Find out why jobs are run in a particular sequence. Identify where a series of jobs really needs to be processed in a certain order, as opposed to an order established arbitrarily.

Constraining – Document conflicts between jobs which prevent them from running together. Identify system resources which may be at the root of these conflicts.

Synchronizing – Prepare a list of checkpoints or external events which may require operator action to trigger or continue processing.

Recovery – Collect recovery procedures and instructions for any jobs which have them.

Once these basic elements are assembled, batch job automation can be accomplished fairly easily and with a likelihood of significant productivity gains.

The Future of Financial Systems on the HP3000

Ronald D. Smirlock
Peat Marwick Main and Co.
2001 M Street, N.W.
Washington, DC 20036

Introduction

One of the most common applications on the HP3000 is that of Financial Systems. Large and small businesses - using all models from the small Series 40 to networked Series 70s - as well as local, state, and federal governmental entities, are using more types of software than ever before to keep track of the ins and outs of cash flow today. Systems covering such specific areas as Stock Market forecasting, Property Value Assessments, Loan Management, and (yes, sadly) Computer Cost Containment programs are in as much demand today as more basic packages such as General Ledger, Accounts Payable, Accounts Receivable, and Fixed Assets Systems.

I. Using the Past to Look at the Future - The Beginning

In any context, you must first look into the past in order to reasonably predict the future. Financial Systems have been around almost as long as computers themselves. After all, computers were created to do high speed mathematics, and accounting involves a lot of mathematical formulas. When businesses were installing their first mainframe systems in the 1950s and 1960s, accounting groups and engineering departments were side by side with their software and computer time requests (in a lot of shops, this continues today). Of the first financial packages, many were strictly devoted to doing elementary tasks, such as totaling and elementary figure comparisons. These systems were very cumbersome to use: typically, all amounts had to be entered onto punch cards, with turnaround times sometimes running into hours. Punch cards were hard to use as well; errors usually were not picked up until after the programs were run, causing the need for additional costly CPU time. Most financial departments could not obtain their own computer systems, forcing them to share computer access with other (and in the eyes of the company, more profitable) departments.

For these reasons, many organizations in the 1960s had accounting groups using paper, pencil, and adding machines as their principle tools. Because of this and the above reasons, many financial departments are still reluctant to use computers as their main tool and do not fully trust computers in helping them solve their accounting problems. In addition, the relationships between many Accounting and Data Processing departments have been (and still are) strained because of those past events. However, events in this area are slowly evolving to the point where these two units are combining to produce some outstanding products.

Spreadsheets and Personal Computers - the 1970s

In the 1970s, the industry saw a great growth in accounting software, mainly due to the growth of more inexpensive, smaller, personal computers. Now, smaller businesses and smaller accounting departments could start to use computers, and even the larger departments could buy their own computers, no longer having to share CPU time with other company units. Many of the first packages developed for these new PCs were financially related. When the establishment of Lotus 123 and other spreadsheets in the computer world occurred, the acceptance of computers in the world of accounting was assured. But what about mainframe computing? Not many companies were willing to invest in large financial systems because of the horrible track record of the genre in the 1960s and early 1970s. Computer manufacturers were being forced to design new hardware to address the specific needs resulting from this problem; that is one of reasons why the HP3000 series came into being.

The HP3000 was (and still is) advertised as a business computer. Many of the features on the HP3000 were designed specifically for financial systems: V/3000 for easier data entry; KSAM (and later IMAGE) for easier record keeping; and the MPE operating system for more English-oriented commands. Other companies later followed with their own "business" machines (even if IBM stands for International Business Machines, they were not exactly on the forefront in this area), but Hewlett Packard's 3000 started the trend.

Financial software evolved in the 1970s as well. Spreadsheets were the program of choice by many accountants by the late 1970s. No longer were those dreaded punch cards or even creating data files necessary. All that needed to be done was to enter the figures directly into the computer, they appeared directly on the screen and almost immediately complicated operations would be performed and an answer given. More and more, paper records were disappearing and computer records were taking their places. Financial systems technology was finally catching up to its scientific brethren.

There were many problems still to be conquered, however. Personal Computers were not the answer for many larger entities, since their record capacities were limited (Remember, in the 1970s fixed disks in PCs were virtually nonexistent). Companies did not want important information existing in boxes of floppy disks that could be easily misplaced. On the other hand, mainframe computers were expensive, and even if such machines as the HP3000 were affordable, new logistical problems would have to be solved. Where do we keep this new, large machine? Where should the terminals be located? Which part(s) of the machine should we secure? How many new people are we going to have to hire to keep the computer in operation and how do we afford that? What about training? Also, who's going to decide what software to purchase? The questions and expenses were so numerous that many larger entities simply stayed with pencil and paper until solutions and money were found.

The DP vs Accounting Wars - the 1980s

Eventually, the solutions and money were found. Many corporations started whole new departments to deal with these problems. Typical names were Financial Systems, Corporate Information Systems, Data Processing, Computer Facilities, and many others. However, they all had one primary function - help the other financial departments with their computer needs. In some cases, I've seen entire computer departments start up on a single day. When these departments started up, many were not subordinate to other financial departments. This was a mistake. These two departments would often run into opposing ideas as to what was the best for their computer system. The DP shop would want a system that was the most compatible with the hardware and software currently running, while the accounting department would want software that had the most features and was the most compatible to its operations. Ninety percent of the organizations I have seen are set up in this fashion. The proper way to establish this relationship is to have both departments work for a single person (hopefully with a background in both computers and accounting). In this way, if the two departments cannot agree on a single approach, this person can make the final decision. This will prevent arguments between the two camps.

Another concern that is coming to the forefront in the 1980's is the problem of computer compatibility. More and more, executives in various organizations have personal computers on top of their desks. However, many companies do not make much of an effort to interconnect their personal computers and their mainframes to share the mainframe's vast financial information. Many of these executives have told me that they do not know why they have spent millions on financial systems when the only output available from the system is on a printout. More people have problems sharing information with mainframe X in their plant, while their other plant has a mainframe Y. They feel bounded by the high technology they bought.

Fortunately, a great amount of hardware and software is being developed and is becoming available today. More about this in the following sections.

What the Future Holds: the 1990s and Beyond

In the coming decades, the emphasis will be on improving the hardware, operating systems and utilities of mainframe computers. Already Hewlett-Packard has made substantial modifications to their database utility system (TurboImage) their operating system (MPE XL) and their hardware (the new HP 9XX Series). Other computer manufacturers have made similar developments, which demonstrates how these makers are desperately trying to keep pace with the growing demands for system capabilities. With the development of more sophisticated software, more sophisticated hardware will be needed to run it. The new Reduced Instruction Set Code (RISC) was developed to allow the Central Processing Unit to run at a higher rate than ever before. With larger amounts of data being stored than ever before, the HP database system, IMAGE, had to be totally reconstructed to hold a larger amount of data elements, data sets, and relationships. Also, with more and more different types of people using a computer than ever before, operating system commands have evolved from "computerese" to the English-like commands of today. In coming years, operating system commands will become voice activated, with individuals who have never used a computer before becoming competent users. For those still using keyboards, command interpreters will become more user-friendly, being able to interpret almost any legitimate sentence.

Now, what about Financial Systems? Recently, Financial Systems have evolved considerably. All financial information can now be entered onto electronic forms painted onto computer terminal screens. In the future, information will be able to be entered through regular, company documents by being read in through a special scanning device. In almost all applications, data processors enter information by typing in what they see written on a sheet in front of them. Why duplicate the information gathering process? Laser scanners would be able to read data from a regular sheet of paper, and store it in the proper place in the financial database. Data entry would become as easy and fast as using a photocopying machine.

Another area for potential growth in financial systems is in data transfer technology. Right now, many good data transfer products are available to communicate information from an HP3000 to any IBM compatible or Apple MacIntosh personal computer. Now, those executives with PCs on their desk can receive up-to-date information into their computers and manipulate it with software with which they are familiar. In addition, personal computer users can do their data entry on their machines, and upload it to their mainframe after they are sure their information is correct. This will help cut down on the use of the mainframe, because many times programs have to be rerun when problems are found in the data. The idea is to eventually use the mainframe simply as a central information gathering point where the CPU's greater speed can be used to great advantage in processing large amounts of information.

Financial systems will be able to take advantage of this new area immediately. Inventory could be taken at local sites, then entered into a personal computer file (such as a spreadsheet), edited for correctness, and finally sent electronically to the mainframe computer. Newly developed cash registers can report sales figures directly from its data center and into the mainframe's database. Satellite offices can send all types of financial information to its company's headquarters for instant analysis. The possibilities are endless!

There are other types of data transfers which can be beneficial in the future. Mainframe to mainframe transfers, where the machines are of different architectures, will become cheaper and more efficient than ever. This will save money in that companies will be able to use their existing hardware while being able to purchase newer machines and keep their flexibility. Companies are increasingly reluctant to keep upgrading their hardware, sometimes even being forced to convert data and programs. Computer manufacturers are now sensitive to this, and most hardware upgrades in the future will no longer require conversions because of planning ahead when building their mainframe systems.

In coming years, the structure of the Financial Systems themselves will change. More and more software vendors are designing their Financial System products to be more complete packages. Instead of purchasing separate packages for General Ledger, Accounts Payable, Accounts Receivable, etc., vendors are providing systems that encompass virtually every conceivable financial need. This takes away the worry of having financial systems that cannot use a common informational source, since all modules will automatically be compatible. The computer operator's time needed for financial system maintenance will decrease, since there are no interconnections necessary between financial systems.

Other financial system areas will improve also. More and more sophisticated report writers are being provided with financial systems. Report Writers are programs that allow users to specify the format, content, and level of summary of various financial information. Report writers are available today, and are becoming more available in other financial software. In the future, virtually all reporting will be user-designed, with other options for report routing, copies, and frequency of reporting.

In the coming years, financial systems will be designed so that software designed by different vendors can still share information. Many accomplish this by providing a format for a standard interface file. In other words, users can format their data into specific structure, and the software will automatically be able to read it into the data file structure and utilize it. This will save the Information System department time in that it will not have to design customized systems and not complicate the data sharing process.

Planning for the Future - How to Purchase a Financial System

Now that the future of financial systems has been discussed, the discussion will now turn to how to choose a proper financial system. The first place to start, however, is not with the software, but with the hardware, since this item is not something likely to be changed. After all, nobody will dare throw out a massive IBM system simply because there exists a fantastic HP financial system. Firstly, look at the available extra disk space you currently have available. If disk space is no problem, you will be able to support a larger software package, and, just as importantly, can hold the source code so it is possible to adjust the system to your specific needs. If disk space is a problem, it might be necessary to buy smaller programs, or even purchase personal computer software and use the mainframe to integrate the data to one location. Also, keep in mind the additional burden the CPU will take. If a system already has over 100 users, complicated, on-line systems (systems that do all their processing at the time data is entered) will bring the system to a screeching halt.

After this evaluation, now is the time to evaluate different financial systems. Most reputable financial software vendors will be happy to provide demonstration copies of the software. If there is room, put many financial systems on your computer at one time. This will provide the capability to simultaneously try similar features without having to remember which software package performed which test. Most importantly, test as many systems as possible! Many companies simply look at the first couple of systems that are compatible with their hardware and budget, and then just select the one that looks best. Vendors will be happy to wait for a while for a decision to be made (if not, the system is probably not worth purchasing).

Next, decide which financial systems are needed. First, write down all the accounting areas the company uses in its day to day business. Typical areas are: General Ledger; Accounts Payable; Accounts Receivable; Fixed Assets; Inventory, and Financial Reporting. There are many other specific areas that also might apply. Next, estimate the amount of man-hours spent in each area and the amount of records generated in each area. Divide the man-hours into the number of records. The system with the lowest ratio of man-hours / records is probably the area which could benefit the most from a computer financial system. The cost of the financial system is also a consideration. Some areas, such as Inventory and Fixed Assets, are much less expensive since the calculations it performs are more straight forward.

Another approach that larger companies sometimes use is to have their programmers create financial software packages. There are many pitfalls to this approach, but it can be done successfully. The first requirement is to form a good team of both data processing and accounting professionals. It is essential that this team be able to work together and that this team report to one and only one upper level manager. After the team is formed, the group should construct an initial design of the particular system. Next, the system should be prototyped on the system. Prototyping is defined by creating a rough, mini-system that simulates the basic structure of both the programming and the database. After the prototyping is completed, the entire group should critique and change the system to better fit the initial design. The eventual users of system should also be allowed to experiment with the prototype and critique it; the more the better. This process should be repeated until the prototype is complete. When finished, the prototype should then be enhanced by adding other parts of the system. Again, the cycle of critiquing and modifying should be used, and repeated until the system is completed. It is important to involve the end users of the system throughout the process to insure the greatest use of the package. Although this process may sound expensive and time-consuming, it is the only way in insure a workable, desirable system that will not be scrapped in a few years.

Since many companies already have existing financial systems, another area of interest is deciding if or when to stop using the current system and obtain a more sophisticated one. Many of the signs can be easily seen. Numerous complaints from the system's users will be heard. The overall computer response time will slow down. Deadlines for reports will not be met. Data errors might occasionally occur. However, there are a lot of other indications of a weak financial system (these questions can also be used to help you select a new system as well):

1. How adequate is the system documentation? Numerous times, infrequent problems will occur, yet when the documentation is used to determine the problem, the manuals prove useless to help.

2. Is there a vendor contact to help you with your problems? Again, if a major problem occurred, having no vendor support could prove disastrous.

3. Does the vendor provide any additional services after the sale, such as the design of additional reports or training. Full service-type vendors tend to be more expensive, but are well worth it in the long run.

4. Does the vendor produce new releases of the financial system that can be obtained at little or no cost. There is little sense in obtaining software that could easily made obsolete, especially in these days of often-changing tax and business laws.

5. Does the system take advantage all available technology that is associated with the computer. It makes little sense to keep an old system that only runs slow because it does not use the latest, timesaving utilities.

Conclusion

The future of Financial Systems is a very complex and ever-changing one. From the punch cards of the 1960s, to the networked, English-responsive computers of tomorrow, financial systems will continue to grow and improve. It is most important to keep up with changes by obtaining a system, whether created in-house or purchased, that can be expanded and modified to keep pace. Given the voluminous amount of systems available, it is not impossible to secure the proper system.

Building your own system is not impossible, either, if the time and effort is made to prototype it, as well as obtaining the input of as many end users as possible. Do not try an add on approach to building a financial system, complete the entire system first, and test it fully before putting it into production.

Do not be afraid to evaluate the current system. If the minor problems are found out first off, larger problems can be avoided in the future. Using the guidelines discussed, an accurate assessment of any system can be made.

Lastly, always remember to find a financial software vendor that is knowledgeable, established, and that will be available after the installation for training and additional modifications and will become necessary later on. Financial systems are always evolving; make sure that the system selected can evolve also.

Management Systems at Westinghouse Furniture Systems:

Total Business Systems Implementation from A Management Perspective

by

Thomas H. Idema

Westinghouse Furniture Systems
4300 36th Street, S.E.
Grand Rapids, Michigan 49518-8829

I. ABSTRACT:

This paper describes the real world successful implementation of a totally integrated business systems plan from the management point of view. It covers manufacturing and office systems integration, systems development philosophy, technical computing in both quality and engineering areas, office automation, networking, data communications and the operational environment. Also discussed are standards for Total Quality measurements which have resulted in a cumulative up time of over 99% for the past eight years despite having grown from a distributed processing operation of one HP3000 Series III to an independent Business Unit which includes both Series 70's and Series 950's.

II. Introduction:

Westinghouse Furniture Systems, Grand Rapids, Michigan, is one of the world's leading producers of office furniture systems. A division of the Westinghouse Electric Corporation, the company represents the "future of the office" with the integration of technology into the workplace as part of its ultra-modern facility which functions as a "living laboratory" for both factory and office productivity. The Westinghouse Furniture Systems facility was dedicated by President Reagan in 1984 where the technologies of the future of the office were demonstrated to be world class and made a technological statement about the integration of technology, people and furniture to create an environment in which excellence can be achieved while maximizing productivity and quality.

We at Westinghouse Furniture Systems have approached systems integration and implementation to make a competitive contribution through the use of technology, but not for it's own sake. Technology is vital to our business as a means for us to operate more productively and to expand the limits of personal performance in a cost effective manner.

We operate in a highly competitive market environment in an attractive industry, but one where product differentiation is sometimes difficult and where execution and quality provide the margin necessary for success. The application of technologically advanced systems has helped us provide this margin and that is why you will see words like, "accuracy, speed, timeliness, flexibility, responsiveness, and accessability" used throughout this paper.

Information Systems are strategic to our business with a mission to provide, "...quality information and administrative services that support Business Unit management in obtaining strategic and operating objectives in a cost effective manner." As such, our approach has been to integrate this mission with the overall strategy of the Business Unit which is "Customer Satisifaction through Total Quality Leadership." While the achievement of this goal is not without cost, we have been able to provide the systems described herein for approximately 1.4% of sales while reaping the generous dividends and payoffs associated with the networking and synergy gained as a result.

III. The Systems:

A. Manufacturing Systems Integration

1. Warehousing

The key to the system development and manufacturing systems integration is, of course, the computer and the development of on-line, interactive, integrated database systems which allow us to manage, control and plan our manufacturing activities. In the warehouses computer systems are utilized to control and optimize the storage of raw materials, purchased parts, and finished goods. Using Just-In-Time (JIT) methods along with various bar code equipment, our systems improve the productivity of material handlers by simplifying warehouse configurations, improving inventory control and optimizing the selection of carriers for shipping.

2. Manufacturing

On-line manufacturing systems, utilizing bar coded manufacturing instruction sheets telling each department what products to produce and in what quantities along with Hewlett Packard on-line data capture equipment, greatly enhances the accuracy of our production efforts. As product is manufactured and packaged, bar code readers and printers are utilized to label all cartons, skids, and pallets and update the manufacturing status of each customer order.

3. Shipping

The computer system automatically keeps track of all customer orders and notifies our shipping department of those orders that have been completed. Bar coded tags are used to stage orders for shipment where the on-line shipping system utilizes bar code equipment to record product as it is loaded onto trailers. The system then automatically produces packing lists, bills of lading, carton content lists, and skid content lists as well as truck routings for more efficient deliveries. All of this paper work, needed by the carriers, is produced on-site in the shipping office and given to the drivers immediately prior to departure.

Once an order is shipped, the invoicing system is updated and appropriate invoicing is produced that same day. With direct interface to the on-line shipping system, the invoicing system provides increased accuracy and the timely generation of invoices and related financial reporting. Automated invoicing also improves the accuracy and productivity of our accounting staff.

Information is also maintained on-line regarding the performance of various carriers and is used to regularly chart an index of carrier quality.

4. Purchasing

The productivity of the purchasing function has been greatly enhanced with the implementation of an integrated purchasing system which allows the Production Planning Department and Materials Planning to directly enter their requirements into the on-line system which is integrated with the manufacturing and materials requirements planning systems (MRP). This along with normal purchasing requirements results in purchase orders being produced automatically for proper quantities and in a timely manner.

The requirement by some vendors for electronic purchase orders has also provided us with the opportunity to realize the advantages and benefits associated with EDI or Electronic Data Interchange. This standard has allowed us to take advantage of economies both internally, within the corporation, as well as with external vendors and selected customers.

5. Manufacturing Planning

Our automated manufacturing systems, (the AMAPS system from MSA), besides providing standard features such as bills of material, standard costing, process and routings, also include automated scheduling in which all orders that have been entered or modified are processed by a program that defines the factory's overall and current capacity and defines material availability. Once processed, the manufacturing and shippping date of an order can be determined and communicated to the customer in the form of laser printed acknowledgments from our HP2680 laser printers. These systems allow us to control costs closely and accurately, keeping the manufacturing process as cost effective as possible.

Computer controlled cutsheet scheduling allows all orders scheduled for a given time frame to be aggregated and optimized for the cutting of bulk stock used in the manufacture of our products. In fact, the NC control for our million dollar automated cutting facility is provided by the same computer program which produces the pattern optimization and layout information.

B. Office Systems Integration

1. Order Entry

In office systems one of the most important applications is that of order entry. On-line order entry personnel utilize HP Vectra's with color screens to enter customer orders. The use of color aids us in the editing and validation of orders in an on-line, interactive manner to ensure order integrity and accuracy. This system provides the basis for the manufacturing and status tracking systems discussed earlier.

In conjunction with this system is a procedure designed to enter orders remotely which takes advantage of both networking and data highways. Remote order entry is designed for use by Westinghouse dealers, selected customers and sales personnel here, in Canada, and the United Kingdom. This application is designed to speed the entry of orders with the use of PC's, specially designed software and Westinghouse communications facilities.

2. Customer Service

Westinghouse Furniture Systems prides itself on customer satisfaction which is part of our strategic direction for competing within our market. As part of accomplishing this goal the Customer Service Department is equipped with on-line instant access to all orders and customer information through the Manufacturing Order Status System which ensures that customer inquiries and questions are handled both quickly and accurately.

For access to archived documents a Computer Aided Retreival or CAR system has also been provided for use within Customer Service Department. The Computer Aided Retreival system is part of the division's Records Management Program and utilizes the fast, accurate database management capabilities of computers and the cost and space-effective technology of micrographics to make information readily available to satisify customer needs.

4. Financial Applications

Financial systems include the standard ledgers and various other applications familiar to each of us from payroll to payables and receivables. These applications are part of our on-line systems as are labor, wage and cost reporting and the time and attendence system which uses Hewlett Packard's data capture time clocks. To facilitate financial control, all Business Unit budgeting and expense reporting is consolidated and reconciled with the use of the Cognos PowerPlan product.

5. Information Center

Our implementation of the Information Center concept is centered around the philosophy of obtaining, testing and qualifying various personal computers and their peripherals for the purpose of being able to recommend the very best configurations to enhance and improve the productivity of those individuals using the equipment. The impact of the personal computer on our organization is such that with our quarterly survey of use and utilization, our measured productivity gains have been so significant that since 1982 we have avoided having to hire approximately one hundred additional white collar staff members. At present our ratio of PC's to white collar employee is just over 1.1 to 1.

The Information Center or more properly called a "Technology Center" is staffed by people responsible for training our personnel in the use of the various hardware and associated software from spreadsheets and databases to graphics and desktop publishing. One of the favorite products of the IC staff was a quarterly newsletter covering all sorts of PC educational and technical subjects called the "TechNickle Journal."

Graphics provide our management reporting, executive and customer presentations with an "impact" that is lost with normal reporting methods. At Westinghouse Furniture Systems, graphics are utilized to prepare analysis of numerical data and production of visual aids for presentations which offer reduced cycle time, lower cost per visual, and less time commitment than manual methods.

6. Marketing Systems

Integrated marketing systems provide for on-line product quotes and discount analysis for our remote sales force using both Hewlett Packard and non-HP portable computers. To provide a timeliness in reporting we have made on-line reports available to our salesforce, including numerous daily, weekly, and monthly reports showing sales performance status, prospects, leads, and negotiation status which they may read and retrieve as they access our systems remotely.

Our Telemarketing department has been responsible for a sizable sales volume as a result of being able to enter product inquiries and potential sales leads directly into a marketing database. This results in sales leads being qualified more effectively. Then, through HPMAIL, both sales and field administrative management are automatically notified of a potential customer interest and in which product areas. In addition, notification and mailing labels are also sent via HPMAIL directly to our Literature Distribution Center telling them which catalog informtion and sales brochures to send to the potential customer.

Another marketing system which has proven invaluable to us is an on-line 24-Hour Response Center through which we can locate and dispatch replacement parts to a customer site anywhere in the world within 24-hours using our integrated materials database.

Our Executive Information System or EIS provides daily on-line marketing and sales reports automatically to key staff and other management personnel. These reports are produced through our normal daily processing and specific copies are then distributed via HPMAIL to the mailboxes of the appropriate managers where they can then access the EIS report and take action as necessary.

C. Office Automation Systems

1. Electronic Mail

Electronic Mail has been a key communication facility within Westinghouse Furniture Systems since 1979. Today local, field, corporate, and international communications are computer stored and forwarded using HPMAIL independent of time, location or individual schedules.

2. Voice Message Exchange

Westinghouse uses Voice Message Exchange or VMX, a computer store and forward voice messaging system for brief, one-way, informational communications and thereby improving the effectiveness of business communications for those people who travel frequently or require confidential communications.

3. Message Center & VISCOM

Two unique systems utilized throughout our facility are VISCOM and our Message-Center. VISCOM, a Visual Paging System which utilizes several strategically places LED screens, is used to visually inform staff, managers, and visiting guests of urgent messages or telephone calls, without audio interruptions in the workplace.

Our Message-Center, a computer controlled telephone support messaging system, is utilized to enhance the organization's image and effectiveness internally and externally. The message center improves communication by ensuring that no telephone message, whether for customer service or normal white collar staff is ever lost because someone may not be available at the time a call is received. This system has provided excellent service, while reducing staff work load and providing secretarial support.

4. HiTech Conference Rooms

All of our conference rooms are provided with a full set of audio visual devices from overhead projectors, VCR tape players, automtically concealed marker boards, special lighting controls, and a unique device called the Data Monitor. The Data Monitor is a conference cost clocking system used to assist in the control of conferences and meetings by computing and visually displaying what the meetings cost based on the hourly salary of the attendees.

5. Central/Distributed Word Processing

All word and document processing from memos to speeches are handled through our integrated Central and Distributed Wang word processing systems which are used to generate high quality documents quickly and accurately through a central automated dictation system. Printing is done with a central IBM 6670 laser printer and in distributed stations with HP Laserjet printers.

Optical Character Recognition (OCR) is also used in our Word Processing Department to directly input text into the word processing system. This increases speed of entry and reduces the chance of errors in the re-keying process for documents that have already be produced in printed form.

7. Teleconferencing

Freeze Frame and Full Motion Television Systems are used in specially designed conference rooms equipped with television cameras, video monitors and control console for reducing the cost and time of meeting with groups of people at off-site locations.

In addition, we have been experimenting with video telephones and remote video cameras which use normal phone lines to transmit video images and allow us to visually as well as verbally communicate with selected individuals and remote sites.

D. Systems Development

Systems development which is responsible for the major systems described herein have been developed largely by a combination of Westinghouse programming staff members and an augmented group of contract programmers which expands or contracts as project demands vary. This mix of employee and contract labor has allowed us to attack various projects and set priorities on a pay-as-you-go basis. This philosophy has also allowed us to justify project funding and schedules based on the needs of the business to implement specific systems projects.

All of our development has been done in COBOL for the simple reason that contract COBOL programmers are a readily available comodity, whereas contractors in other languages, or so-called third and fourth generation languages are not so accessable. On the other hand, Westinghouse has taken full advantage of the capabilities offered with integrated database systems using IMAGE, and of course, utilities such as those offered by Adager, Robelle, VESoft, and DISC's OMNIDEX and IMSAM. In addition to these packages, we also use HP's REPORT and INFORM for user ad hoc reporting and have done some work with Gateway System's package called Synergist to develop PC/Mainframe distributed applications, although this has not proven to us to be the way of the future just yet.

E. Technical Computing

Our Technical Computing Department is responsible for Engineering CAD/CAE functions which currently utilizes Tektronix workstations networked to dual HP 9000 model 550 computers. Anvil/5000 software is utilized to do all of our drafting and engineering design work. A series of large scale D and E size HP plotters and 3M aperature card production facilities are used for hard copy reproduction of computer stored drawings.

In the near future this will have been replaced by a series of HP 330 engineering work stations networked to an HP 9000 model 835 precision architecture computer using HP-UX.

One of the biggest benefits of using the technical computing arrangement that we have is the flexibility available which allows us to not only network to our HP 3000 business computers for bill of material updates but also to feed drawings directly to the manufacturing floor via on-line CRT's and to be able to feed numerical control information to various equipment for limited aspects of Computer Integrated Manufacturing (CIM), using IGES standards.

These technical computing systems also interface to our Quality Assurance systems to facilitate the design, development and testing of our products, assuring their quality and reliability.

F. Quality Assurance

Quality Assurance systems include an HP 1000 model A700 which, along with a third HP 9000 model 550, is used to administer our Statistical Process Control (SPC) programs. In addition, various other equipment, including HP 9816's and Vectra PC's are used to run instrumentation and color spectrometers throughout our manufacturing areas. Voice data entry systems, of a Westinghouse design and manufacture, are used with Quality Assurance systems where terminals and other data input devices are not readily available.

G. Marketing CADD

Wes-CADD/PC is a Westinghouse developed computer-aided drafting and design system producing full color two-dimensional designs which was built and expanded using AUTOCAD(TM). The System is utilized by clients, dealers and designers for systems furniture layout. Repetitive, time-consuming tasks are performed by the CADD system allowing both manager and designer to explore several alternatives to layouts in less time than it used to take to explore just one. In addition, bills of material can be automatically generated and others entered directly from this CADD system into the order entry system. Add on options include the Westinghouse Quick Quote and Remote Order Entry systems which directly interface to other Business Unit systems.

H. Networking/Communications

Where the use of database technology has been the key to the integration and overall cohesiveness of our total system approach, our networking and communications facilities have provided the structural backbone for this implementation and allowed us to construct and use data highways to link the vital elements of our business.

Externally, we have made extensive use of Westinghouse corporate facilities by linking our X.25 field support network with the Westinghouse packet switching network called WESPAC to achieve an economy of scale. Likewise, it is hoped in the near future that we will be able to use similiar facilities to link our HPMAIL to the corporate electronic mail system using X.400 protocols as part of the ISO standard.

In order to make the most effective use of the physical link between various corporate data centers and other facilities, we use an INFOTRON 790 Statistical Multiplexer with leased 19.2 KB and T1 56KB lines along with a MICOM Protocol Converter which virtually lets anything at our facility talk to anything at the other end of the line. Imagine an HP 150 talking to a large IBM mainframe computer interactively with no special software and neither one realizing that the other device is not what they think it is.

Internally, all of our terminals and PC's are connected to our computer hardware via a MICOM Instanet 6600 data switch which allows us the flexibility of having any user choose which computer, (we have five), they need to be connected to for a given application.

Although our mainframe computer systems are physically networked using the HP 802.3 Local Area Network and use NS/3000, the real flexibility we have experienced has been with our Allen-Bradley broadband coaxial Local Area Network which is used to link all Business Unit personal computers to each other, to the main computer complex and to external electronic communication networks, maximizing the amount of information an employee can receive through the use of a personal computer.

Not only are the networked PC's able to share data and peripherals as a result of the broadband LAN, but we are also able to use some of the bandwidth for video which allows us to transmit from our factory or anywhere else within our complex to conference rooms or to our teleconferencing facility, live broadcasts of operations or special demonstrations.

I. Operational Environment

Our Data Center consists of three HP 3000 Series 70's and two Series 950 computers, twenty-seven Eagle disc drives and 14 7933 disc's, four 7980 tape drives and two 2680 laser printers. No third party computer hardware is used in our data center. The computer room is environmentally controlled, powered by an Emmerson 256 KVA Uninterruptible Power Supply (UPS) system, and protected by a Halon 1301 fire suppression system.

Within the Data Center our goal is to provide Total Quality Service to the Business Unit on a twenty-four hour a day basis as measured by availability, response time, and uptime. These three measures are the key elements in being able to evaluate the level and quality of service provided to our user community. Availability means that 90 percent of the time the on-line systems must be up by seven each morning; response time must average less than 2.3 seconds and uptime must be greater than 99.6 percent. Happily, we have been able to exceed these goals consistantly over the past several years. In 1987 availability was 97 percent, response time was 2.2 seconds and up time was a remarkable 99.99 percent for the year. When you consider that these measures are for all machines over a 24 hour day, five days a week, (although we operate seven days a week, more often than not), they become all the more phenominal.

Other Data Center statistics are equally amazing. For instance, in averaging over 39 million lines of print our HP 2680 laser printers go through two and a half tons of paper each month...enough continuous form paper to stretch from Grand Rapids, Michigan to Orlando, Florida. These print lines do not include the thousands of pages of micro fische (COM) output that is produced each month or the print produced at our twenty-seven remote printers for product labels and wharehouse tags.

Our Data Center is managed by a highly trained professional staff consisting of one lead and three additional computer operators using excellent written procedures and Unison's Maestro and Spoolmate products. We also have a System Manager who is responsible for all systems and systems software, a Data Base Administrator who is responsible for all aspects of our data, it's integrity and recovery, an Electronics Technician who is responsible for the maintenance and repair of all of our terminals, PC's and related peripheral equipment, and a Senior Project Leader who is responsible for all of our communications and technical computing.

A year ago we experimented with the use of a HELP Desk to provide a focal point within MIS to channel all user requests for service and assistance. Although this test was well received and the results excellent, (both in concept and execution), we had to end it for lack of personnel to staff the operation adequately.

IV. Conclusion

The foregoing has covered a great number of subjects in a necessarily brief manner, but from a management standpoint it has had much to do with our competitive success, particularly in the areas of Total Quality Leadership and Customer Satisifaction. Systems reliability together with the interactive and integrated nature of our systems, including the success we have had in integrating the technologies, have been key elements in this overall achievement.

In this success, Hewlett Packard, as a company, has played no small role in that they have taken both a participative and proactive stance as a member of our team in making much of this possible. In addition, they have continued to provide us with a constant stream of new technologies from laser printers to precision architecture computers to help us sustain our successes.

As a result we have accrued numerous benefits including being able to maximize customer satisifaction, achieve a more timely reporting of business critical data, with more accurate information and increased integrity. Among our personnel we have measured improved productivity and gained a true sense of job satisifaction. Our staffs are more knowledgable and information literate and we have seen better and more timely communication among our business professionals. And, we have been able to reduce our costs in both analyzing and communicating information.

Industry trends show that there is an accelerating use of computer and computer related technologies to improve the productivity of the white collar worker. There will undoubtedly be an ever increasing need to integrate these new office technologies into the office environment along with better ways to coordinate the need for information. Any technology that will reduce or eliminate the need for human intervention can successfully be utilized to support the business professional and permit the white collar or "knowledge worker" to perform those tasks that are uniquely human...think, plan, and innovate. This is the goal of the systems organization at Westinghouse Furniture Systems.

Biographical Data:

Thomas H. Idema

The Manager of MIS Technology Services for the Furniture Systems Division of the Westinghouse Electric Corporation since 1980, he has worked for both General Foods and Hewlett Packard since serving in the U.S. Marine Corps where he flew jets and served in the Vietnam War.

Tom graduated with a Bachelor of Public Administration degree with a major in City Management from the University of Mississippi. He obtained his Masters of Business Administration with a major in Management from Western Michigan University.

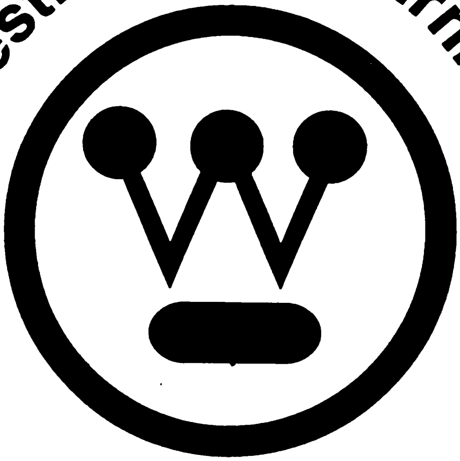
He has over eighteen years of management experience and has taught college level courses in both the systems and management fields for over fourteen years. Tom is a Certified Systems Professional and has been active in the systems field as a professional member of the Association for Systems Management and the Association for Computer Operations Management. In addition, he has been involved in the user group activities of INTEREX, the International Association of Hewlett Packard Computer Users, and is a past Member of the Board of Directors of that organization.

Westinghouse Furniture Systems

Westinghouse Electric Corporation is made up of twenty three strategic Business Units and is one of the world's largest producers of power machinery and equipment. The mega-corporation manufactures a wide range of products from Micarta, a hard surface laminate, to atomic reactors that drive the Navy's nuclear submarines.

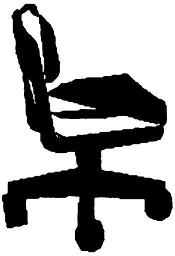
Westinghouse Furniture Systems, one of the leading producers within the office systems industry, is dedicated to the pursuit of product excellence and efficiency through technological innovation. The centerpiece of this commitment is the "Westinghouse workplace," located in Grand Rapids, Michigan, which is a living laboratory, showcasing Westinghouse's complete office product line as it is used in a daily working environment by the employees. This "office of the future" and associated "factory of the future" heralded a new dimension in total workplace productivity that prompted President Ronald Reagan and former president Gerald R. Ford to personally preside over the dedication ceremonies in September, 1984.

Westinghouse Furniture



Systems

Characteristics of Business



- Highly Competitive
- Attractive Industry
- Differentiation Difficult
- Quality / Execution

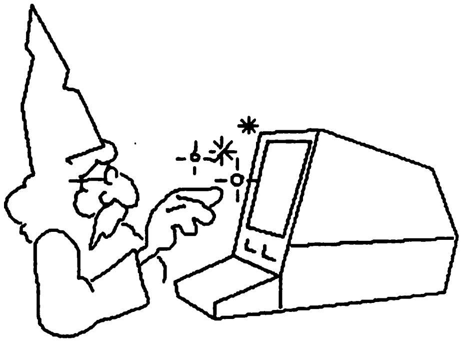
Management Systems at

Westinghouse Furniture Systems

Total Business Systems Implementation From

A Management Perspective

Information Systems:

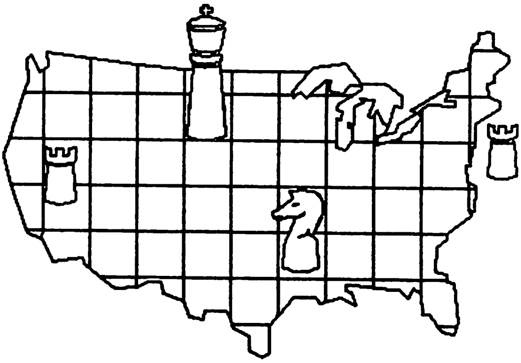


Mission:

To Provide quality information and administrative services that support Business Unit management in obtaining Strategic and Operating Objectives in a cost effective manner.

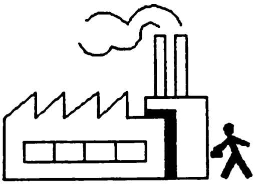
Information Systems:

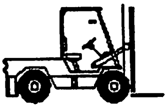
Strategy:



Total Quality Leadership

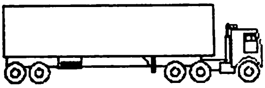
Manufacturing Systems Integration:

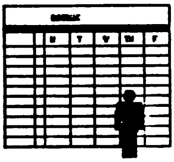




- **Warehousing**
- **Manufacturing**

- **Shipping**
- **Purchasing**

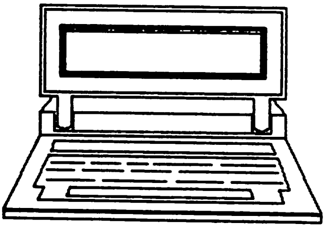




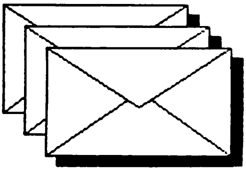
- **Manufacturing Planning**

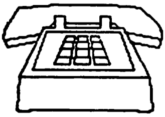
Office Systems Integration:

- Order Entry
- Customer Service
- Financial Applications
- Information Center
- Marketing Systems

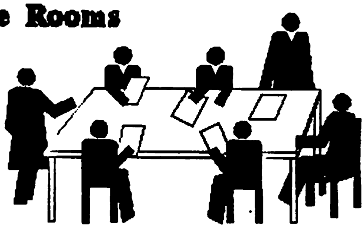


Office Automation Systems:



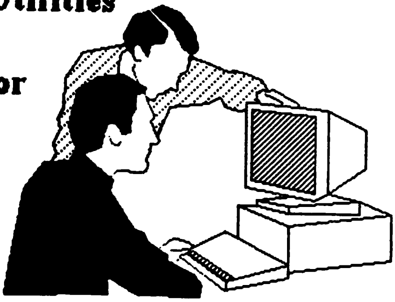


- Electronic Mail
- Voice Message Exchange
- Message Center & VISCOM
- HiTech Conference Rooms
- Word Processing
- Teleconferencing



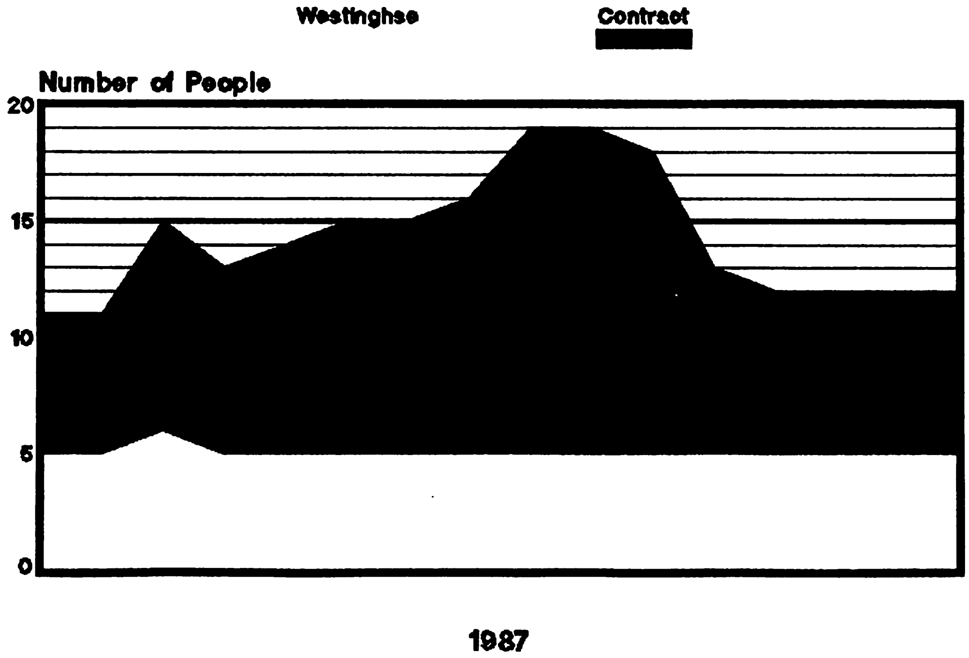
Systems Development:

- Image Based COBOL Systems
- Third Party Software
- Third Party Utilities
- Contract Labor



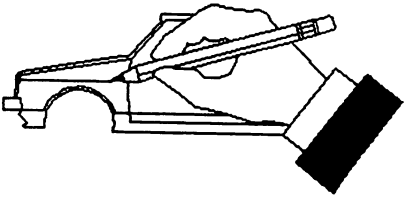
Westinghouse Furniture Systems

MIS Programming Staff by Type



Technical Computing:

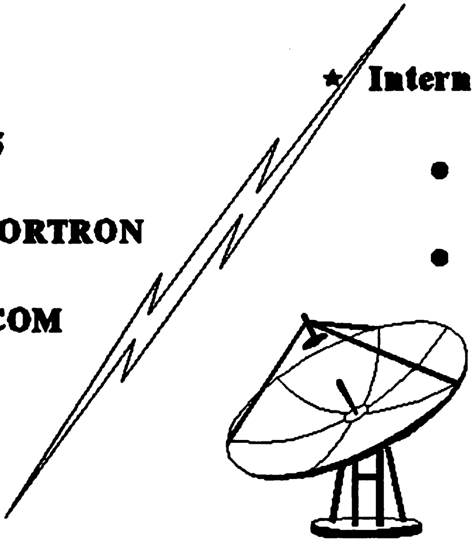
- **Engineering CAD/CAM**
- **Quality Assurance**
- **Marketing CADD**



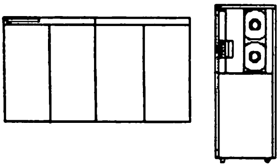
Networking / Communications:

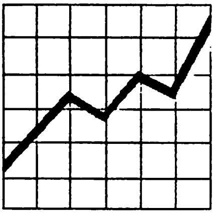
- **External**
  - **X.25**
  - **INFORTRON**
  - **MICOM**

- **Internal**
  - **MICOM/PC**
  - **Two LAN's**

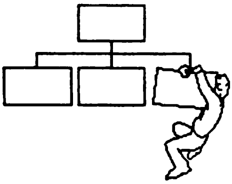


Operational Environment:



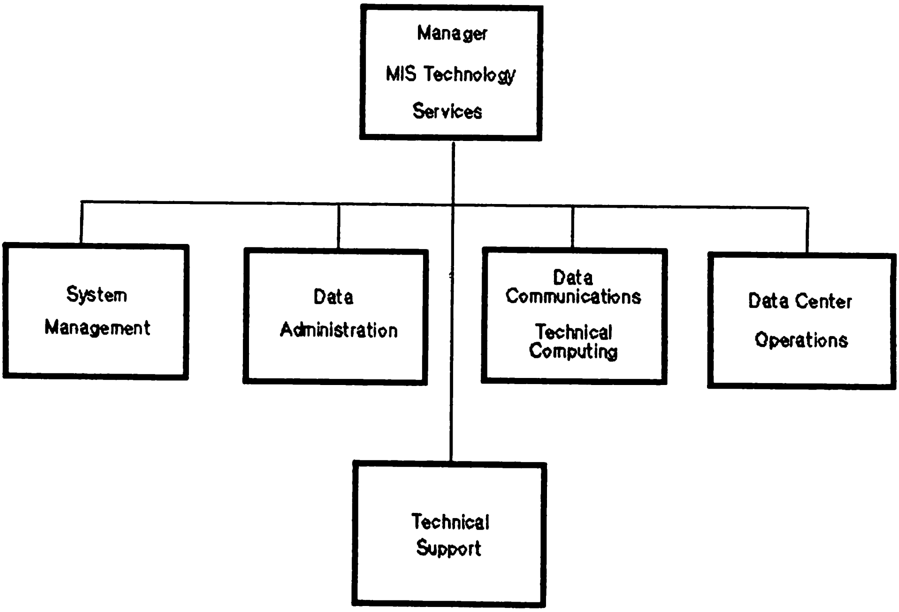


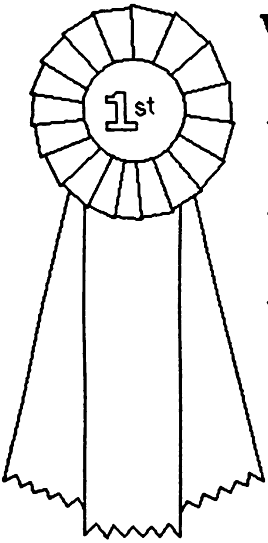
- **Equipment**
- **TQL Goals**
- **Staff**
- **Help Desk**



Westinghouse Furniture Systems

MIS Technology Services Organization

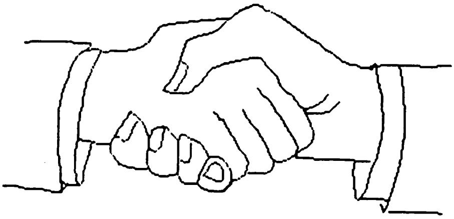

Manager MIS Technology Services
System Management
Data Administration
Data Communications Technical Computing
Data Center Operations
Technical Support



What Role Has All of This Played?

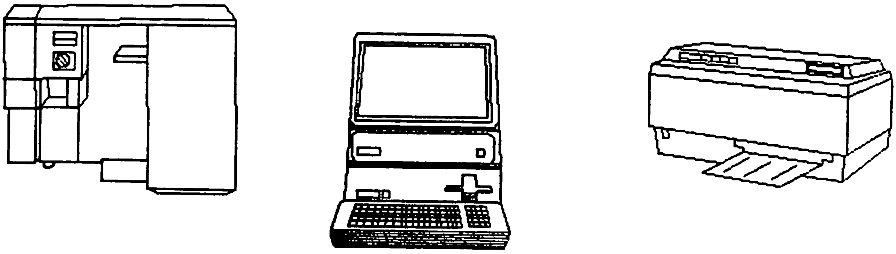
- Reliabillity, Productivity, Quallity
- Interactive, Integrated Systems
- Integration of Technologies

Hewlett Packard's Role:



- ★ **Participative & Proactive**

- ★ **Constant Stream of New Technologies**



BENEFITS:

- Maximize Customer Satisifaction
- More Timely Reporting
- More Accurate Information
- Increased Information Integrity
- Improved Productivity & Job Satisifaction
- Staffs More Knowledgable & Information Literate
- Better More Timely Communication
- Reduced Costs

TITLE: Looking at HP's Telesup

AUTHOR: Isaac Blake

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 0158

TITLE: Database Application Development Using Pascal

AUTHOR: Giles F. Lewis

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 0160

PC PowerHouse : When and How?

Abstract of a paper for presentation
at the Interex HP3000 Users
Conference at Marriott World Center
August 7-12, 1988

by

Suzanne Harmon
AH Computer Services, Inc.
8210 Terrace Drive,
El Cerrito, CA 94530-3059
(415) 525-5070

Users of Cognos' PowerHouse product have been waiting years for a PC product. Now that the product is available, how do the pricing, hardware requirements, functionality and performance affect when we will use the product. When we do decide to use the product for a specific requirement, how do we make it work, what are the perils and pitfalls?

This paper will focus on deciding to use the product and making it work once the decision is made. Its merits and downfalls are discussed as they pertain to:

- stand alone applications
- emergency PC level back up for HP3000 applications
- PC processors of HP3000 based data
- development work stations

Using the product will focus on the specific differences this user found in each of the following products: Dictionary, Quick, Qdesign, Quiz and QTP.

I've been waiting almost three years for Powerhouse to be available on PC's. During that time Speedware, Powerhouse's biggest 4GL competitor on the Hewlett-Packard, Oracle, an important development contender, Synergist, and other products have been, or become, available on the PC. Last fall our company began BETA testing the PC Powerhouse product. The product was released for general sale early this spring.

Many companies have come to see PC's as major "players" in their overall automation strategies. Their use until now has been primarily relegated to office automation, spread sheet analysis, and allowing certain maverick departments to design their own stop-gap applications using PC based tools such as DBaseIII. The advent of major 4GL's and development products, already available on mini and mega-mini machines, porting their products to PC's dramatically changes the choices available to today's businesses.

Our company does a substantial percentage of system development using Powerhouse. For us, a PC version of the product would give us the following options:

Developer's Workstations - Our telephone expenses directly attributable to dial-up to client's HP3000's for development runs $2000 to $3000 a month. We are also at the mercy of the client's back-up schedules, peak production periods, and so on, which may severely impact our ability to put in the time we need to meet our commitments on a project. Therefore, we have had high expectations of developing and testing systems or pieces of systems on the PC and then porting them to the HP3000 for final testing and implementation.

Host Computer Extension - Many applications have functions which are very data entry intensive, or which must be up and running at all times if the company is to remain open for business. We have often wished we could run such pieces of a system on PC's, or at least have that as an option during busy periods, or at times when the HP3000 was down either expectedly or unexpectedly.

Standalone PC Applications - We have an application which we sell as a package, which many potential customers have wanted for years, but their budgets would not enable them to purchase the requisite HP3000, though almost all of them have PC's or can easily obtain them. We have been very eager to release this system as a PC package, and in fact it is this system which we converted to the PC for one of our BETA tests. We also have, as I'm sure all of you do too, many ideas for easy to develop systems which would be much in demand if they were available on PC's.

More recently, a whole new trend in the industry has made me consider a fourth option as possibly the most exciting of all. We have many clients who have offices, or regions, or divisions who all need the same system, but who vary dramatically in size. Where one site may justify a model 58 or 70, another may barely justify a PC. The idea of being able to develop one application system for the client with the assurance that he can install it on a PC, an HP3000 or a Spectrum opens up a new world of possible configurations for cost effective distributed processing.

With these goals in mind, we enthusiasticly welcomed the arrival of PC Powerhouse. The first issue in our minds was: What is the minimum hardware I need to use this product (When you read this paper, please keep in mind that it had to be written and submitted a full three and a half months before Interex, so I cannot be assured that what is true when I am writing it will still be true when you are reading it). Cognos lists the hardware requirements as:

IBM PC-AT or 100% Compatible
- DOS version 3.2 or later
- 640K RAM
- 20MB hard drive (PC Powerhouse occupies 5MB)

IBM PSII Model 50 and up
- OS/2 version 1.0
- 2.5MB RAM

Optional performance enhancements
- 1MB or more expanded memory

We have tried the product on a variety of IBM PC-AT Compatibles, both 286 and 386. The configuration requirements as stated are accurate. However, we did find a couple of things worth mentioning. On one of our 386 machines, we had 512K of main memory and 2MB of extended memory, which is being managed using the Deskview memory manager product. Though this allegedly makes our machine "look like" it has 640K of main memory, in addition to the 60K used by DOS, 40K is used by the memory manager. Powerhouse requires that the entire remaining 580K (after DOS) be available for its use. Because Deskview is using 40K, only 540K is available to Powerhouse, and Powerhouse will not run on this machine. There are allegedly two solutions which we have not yet tried. The first is to buy an "Above board" of 128K or more to expand the main memory without the use of a memory manager. The second is that supposedly using Windows as the memory manager will enable us to run with the present hardware configuration. Though the product will certainly run with a 20MB hard drive, I would not recommend that you seriously consider doing anything with less than a 40MB hard drive. We filled up a 20MB hard drive very quickly and ended up having to clean-up and swap stuff to floppies constantly. We also feel that investing in the fastest disc drive you can find will pay off if you intend to use the product very extensively, as waiting for disc IO will come as a major shock to anyone who is used to working on an HP3000, however slow and overloaded. The newest release we have says you can put your temp files on virtual disc (in memory) to speed things up.

We have not tried the product on a PSII. We have also not tried the relatively recent enhancement allowing you to enhance performance with the automatic use of expanded memory. This is reportedly a rather major performance improvement.

What we found, was that a "reasonable configuration" which would run the product will cost a minimum of about $2500. This is assuming that you feel, as I do, that buying an AT which doesn't have a color monitor is sort of like buying a Rolls Royce with vinyl upholstery. However, this price does not include a super fast drive, expanded memory, a printer, and so on. To really create a developer's workstation would probably run a minimum of $4000.

Now that we have the hardware, what's next? Well, PC Powerhouse isn't cheap. The pricing for 1 - 4 copies is $1295 per copy plus shipping. For 5 - 24 copies it goes down to $995 per copy plus shipping, and so on. This price includes a very basic version of Reflections, which allows you to do terminal emulation and file transfers. It does not include an editor, even though the "main menu" has an exit to editor as one of your options. This is to allow you to create an interface to the editor of your choice. Run time versions of the product are available, so that once the developer has created the application, not every user must fork over the full development copy price. Run time (with or without reporting) must be bought in "minimum lots" of 25 copies (with reporting) or 50 copies (without reporting), with a pricetag of $395 per unit (total $9875) or $150 per unit (total $7500) respectively.

We were incredibly impressed from the outset with the attention that had been given to the ability to port dictionaries, code and data from the HP3000 to the PC. However, the entire structure of Powerhouse use on the PC is much closer to Powerhouse on the VAX than to Powerhouse on the HP3000. Discussing this briefly will help with understanding the "port" process.

File Structures - HP3000, or host files, are not directly accessible by PC Powerhouse, the way they are from the Synergist for example. You must create subfiles from the HP3000 and pull these down to the PC. On the PC you do not have Image or Image-like file structures with PC Powerhouse. You have the equivalent of MPE files and KSAM files.

File extensions versus Groups - On the HP3000 we can keep our source or compiled Powerhouse code wherever we want, and every system seems to have a different idea of what group names to use. When Powerhouse ported to the VAX, they standardized to fixed file extensions, which has carried over to the PC product. All Quick source is in .QKS and all compiled Quick is in .QKC and so on. Therefore, within your source code, you may not include file extensions such as "RUN MYSCREEN.QKS."

Menu Driven - though you can override this and run Powerhouse programs directly from DOS, Powerhouse can also be used as an entirely menu driven process by entering PH at the prompt.

Automatic conditional compile recognition - There is now some standard "recognized" conditional compile syntax. If @IF PC is used, and Qdesign executes in PC Powerhouse, it will automatically recognize the condition as true. Similarly, if @IF HPMPE is used, Qdesign executing on an HP3000 MPE based machine will recognize the condition as true. Thus, one set of programs which can run on both the HP3000 and the PC can be maintained by simply making conditional that code which must differ between the systems. This is true from the dictionary through all of the "programming" products.

Obsolete syntax - For those of you who still have programs which use "string" instead of "character" (like me), and other obsolete syntax from versions prior to 5.xx, all such code must be removed prior to porting. Obsolete syntax is strictly not allowed.

Security - Since PC Powerhouse is not yet a multi-user product, there is no security syntax such as users, locking, and so on. This code must be omitted from the PC version of programs.

Blockmode - Yeah!!! Blockmode is not available in PC Powerhouse.

Items - Due to storage differences between the PC and the HP of certain types of data items, you must be careful to specify size at the element level rather than the item level and to be careful in redefine situations.

Subfiles - Subfiles on the PC are always permanent, whereas on the HP they are temporary unless specified otherwise. Your disc space can fill up very quickly with subfiles you thought were temporary on the PC.

Temporary Files - Beware of temporary files (Powerhouse sort and work files) which are normally deleted at the end of a Powerhouse session on the PC, but will not be deleted if there is an abnormal interruption of Powerhouse. These, too, will fill up your disc space.

Some of the features which are available on the PC are extremely exciting in terms of creating sophisticated and appealing applications. A few of my favorites are:

Color Usage - Color can be used for hiliting, line drawing, background, data, etc. The combinations are mind-boggling and can be aesthetically pleasing or nauseating, as you choose.

Windowed Screens - A called screen can overlay the calling screen in any window you choose, for any length, width and location. You can stack these on top of each other to the point that the screen looks something like all the papers on my desk overlaying each other and peaking out here and there.

Field Editting - For lack of a better description, you can now type over the data in a field the way you always could in block mode, rather than having the current data disappear before you can enter your changed data.

Scrolling Fields - One of my very favorites! If you have a very full screen and a 40 byte name field, for example, you can display 20 bytes of it and scroll the data left and right. Quick provides a little asterisk beside the field to let you know there's more.

Before discussing performance, I have to confess to you that I am a very neophyte PC user. Therefore, my expectations were that by having this whole machine to myself, with 640K of main memory and 20MB of hard disc, things were going to fly! If you are not a neophyte PC user, then you know that I had a rude shock awaiting me. Besides the fact that PCs are slow compared to even heavily loaded HP3000s, the early beta versions of the PC product were very slow. Cognos has spruced up the performance by many magnitudes, and as a PC product it now performs respectably.

Given hardware requirements, functionality, price and performance let's review my current thinking on the various uses the product might have.

Developer's Workstation - Our general feeling after a couple of attempts to use the product for this purpose is thumbs down. First of all, equiping a developer's workstation for use with PC Powerhouse is far more costly than a developer's workstation to be used as a terminal to the host via Reflections or whatever. Second, if you have a tendency to flip back and forth a lot between an editor and Qdesign, which our staff tends to do, you spend half your life waiting for the editor to load or Qdesign to load. Actually, we felt it took 3 to 4 times as long for these functions to load. Third, we use a lot of "tricks" in Powerhouse, which we feel give the product an enormous amount of flexibility and versatility, such as using Quick to write Quiz programs based on user input. Many of these things can't be done in PC Powerhouse due to the limitations of DOS. Therefore we found ourselves only being able to use the PC for a limited subset of the development. We also do a lot of development with Omnidex, which can't be tested on the PC. Our general feeling was that if you are using the product to get up a relatively easy application or prototype, it would probably be great, but for the kind of development we do it was too time consuming and too limited.

Host Computer Extension - As a data gatherer, PC Powerhouse would be great! You wouldn't get caught up in programs loading because you could stay inside Quick the entire time. The user could madly enter data all day, which could be easily uploaded at night, using far more attractive screens than are available on the host. It would also be ideal for allowing user reporting from data extracts downloaded from the host.

Standalone PC Applications - This is probably tied for ideal uses for this product. If a user is strictly a PC user, the performance issues which might drive me crazy will seem perfectly normal to them. A client of mine recently demoed one of their applications to me which runs on a PC (not Powerhouse). I couldn't believe how long it took for screens to load, etc. Compared to that, PC Powerhouse looks like a Grand Prix winner. Furthermore, all the advantages of an HP3000 application developed in Powerhouse, such as ease of maintainability and short development cycles, will also be true on a PC.

Distributed System Flexibility Maximizer - The value of PC Powerhouse for this use depends a lot on the application. If the application is so complex that it falls victim to many of the problems inherent in using the product for a development workstation, then it is likely that such a large amount of the system will have to be developed and maintained individually for the PC and the host that any possible benefit will be lost. However, if less than 20% of the system would be different for the two versions, it could be a very major benefit to use this approach. Coming out with a remote file access capability could also change the appeal of this option.

In general, the product was a mixed bag of exceeding my expectations in some areas and not meeting them in others. I think we will use the product extensively, but I think we will approach it more cautiously that I would have projected a year ago.

Effectively Understanding User Requirements

by

Suzanne Harmon
AH Computer Services, Inc.
8210 Terrace Drive
El Cerrito, Ca 94530
(415) 525-5070

Though I don't have time to read the daily comics as often as I would like, I really enjoy Mr. Boffo, and especially the occassional "Unclear on the Concept." As I was laying awake one night recently, I began pondering my commitment to produce this paper, and my mind was wandering the fields of several design efforts of years past as examples of do's and don'ts.

One, especially, came to mind. Picture a medium sized organization where several distinct operational departments, in addition to Finance and Sales and Marketing, have no visible (or invisible) means of communication. Policies set by one department are rarely even known by another. Customers call regarding advertised specials, which order takers have no knowledge of, or ability to enter in the system. Due to this lack of communication in combination with inadequate and/or non-existent management information systems, the organization appears to be loosing about $750,000 a month more than they need to simply due to penalties, etc.

We were brought in to do a requirements definition which could in turn be used as the basis for an RFP (request for proposal). After going through the design process and publishing and distributing the requirements document, I called the head of MIS, who was my liaison, to ask what feedback she had had and when we could schedule a walkthru of the document. She reported that the users were extremely disappointed in the document I had produced. They felt that "all it really did was present what they already knew they wanted and had asked for in the course of our interviews," and that "anyone could have come in and documented how they did business, and what they wanted the system to do." As I reminisced there in the dark, I could not help but giggle at the thought of overlaying the picture with an "Unclear on the Concept" heading.

Despite my amusement at what I feel was an obviously comical situation, I had signals from the very beginning of that effort that it would end badly. If we went through a video of the activities which took place, you would be able to check off a violation of virtually every point as we discuss "Effectively Understanding User Requirements," and the steps to insure the success of this process.

Setting Expectations Up Front

Whether the "User Requirements" process will take two days or two months, the first step must always be the mutual understanding of how the process will work and setting realistic expectations. What I have found to be the most effective way of doing this is to have a "kick-off" meeting which should be attended by the department or area head and their most knowledgeable designee (if possible) from every department or area which will be a direct or peripheral user of the system. At this meeting you want to tell the attendees:

1. Who you are. Provide not only your name, but also your background. Be up front with them about your strengths (ie. I have designed five major systems which are still successfully in operation) and your weaknesses (ie. I have never designed a widget tracking system before, so I will be looking to each of you to help me better understand how this business works).

2. Why you are here. Make this as strong a statement as possible, as this can dramatically affect the co-operation you will receive (ie. "The president has determined that a widget tracking system will increase our profitability by 15% and is looking to all of us to make this happen as quickly and successfully as possible," would be more effective than "I've been given the job of designing a widget tracking system and I need your help.") If possible enlist someone high up in the company to do the initial introduction, someone who will command the attention and response of those present.

3. What you expect to produce. Lay out for the users, in as much detail as possible, what the end result of this effort will be. If you have examples of previous similar efforts, use them to illustrate your points. What the users can expect should be something which will be meaningful to them, in their language, not a bunch of data processing mumbo jumbo. Data flow diagrams and the like may be fine

for a systems analyst who has been to a structured design class, but they mean zilch to a payroll clerk.

4. What you will want from them. The first thing users always want to know is "how much of my time is this going to take?" Remember that whereas this is your job, they have to do their jobs above and beyond the time they spend with you. However, this is a difficult question to answer, and of course can vary depending on how good a resource they turn out to be. It also varies by system, as some systems take weeks of interviewing while others take only hours. For an average system I would typically respond something like "I will want to spend two to three hours with each of you initially to become familiar with your area. At the end of that meeting I can give you a better idea of how many sessions we will need beyond that. Usually, I will ask you to spend one to two more three or four hour sessions with me. After that I can usually call you with questions."

5. How their time with you will be spent. I like to prepare them for the time we will spend together, so they can think about what they want in advance, and so that they can begin to gather the materials they use in their job and pay attention to some of the details of their job they may take for granted, but which will be important details for me. I describe the interviewing process, the kind of documents I will ask them for, and the kinds of questions I will ask. I emphasize that though I know they are extremely busy, interruptions, meetings cut short, and so on can severely impact the successful outcome of our time together. That way they can plan in advance to have someone else field problems while we meet.

Once this part of the meeting is complete, you want to do two very important things:

1. Let them know that you will personally reconvene with the entire group and go through your ideas and understandings before producing the final document. This is extremely important, both to the process as a whole, and in terms of taking some of the pressure off them to be "perfect" during the interviewing process. This lets them know they will have a chance to catch their mistakes as well as those of others before it is too late.

2. Ask them if they have questions, concerns or problems with anything you have said. If they want to negotiate anything, up to and including the format of the final document you will produce for them, now is the time to discuss it. Remind them that the goal of this meeting is to make sure that everyone leaves the room with the same goals and the same expectations for this process.

Effective Interviewing

The interviewing process is like an art. It requires talent, skill, practice, the right equipment, discipline and patience.

1. Timing and Environment.
   User interviewing is very intense. You should attempt to limit user interviewing to four hours a day so that you don't spend time when you are too tired or burned out to be effective. Mornings are best because frequently people get a bit logy in the afternoon. Avoid spending less than two hours, because you can't get enough completion or depth in less time.
   A small room with a door that closes, no phone and good ventilation is best. The user should have a feeling of intimacy and privacy, so that they will not hold back information that might be important. Poor ventilation can make people drowsy. A closed door and no phone help prevent interruptions.

2. Equipment.
   There are a few things I consider essential for user interviewing. First and foremost is a tape recorder and lots of blank tape (90 minute tapes for infrequent changes). I assure the user that anything they say will be kept in the strictest confidence and that no one else will ever hear the tapes, but that they help me go back later and verify my recollections of our conversations. If the user resists heavily, drop it, but otherwise be diligent about recording every word - it's invaluable. Second, have plenty of pads of paper and pens or pencils. Even though you are recording every word, you will want to jot down key points, notes to yourself, questions which you want to ask but don't want to interrupt, and ideas. Besides, people think you are more attentive to what they are saying when you take notes. Third, have a flip chart with colored pens to use for illustrating key ideas. Use different colors for different functions or departments. Flip charts are much better than boards, as you can tape up many sheets around the room to refer to as you are talking (don't forget the masking tape), and then take them with you when you are finished for future reference.

3. Control
   One of the most critical, and most difficult, aspects of user interviewing is to make sure that you are the person in control at all times. If you lose control of the process you can come out of the interview having wasted your time and everyone elses, with no solid information to show for your efforts. Maintaining control of the user interview means that you direct the conversation and insure that what is being discussed is precisely what you want discussed at any given point, that the level of detail progresses to the level of detail you need, and that you

are getting "real" information. There are three classic things that I have found I need to guard against to maintain control. The first is when the user I'm interviewing wants to tell me about their design for the system. Many users fancy themselves better at designing systems than the people interviewing them (and probably some of them are). I respond to this by saying "Gee, those are some great ideas, but it's too early for me too even be thinking about those things. Before we get into all that I need to understand ...." The second, which is even more delicate, is when someone from my clients MIS department wants to tag along so "they can understand the requirements too." Then, either to impress me or to impress the user, I'm never quite sure which, they start monopolizing the conversation. They either start on a diatribe of how the user's department functions, or start asking their own questions, which are rarely the same questions I would ask. How you handle this one depends totally on the politics of the situation. My only advice is, if you can't get them to shut up then end the interview as quickly as possible (ie. develop a terrible case of stomach flu). Third, is the situation where you are meeting with the department head and their designee, and it quickly becomes obvious that though the designee knows everything and the department head has no idea how things operate, the department head either wants to do all the talking or has the designee so brow-beaten that they won't give you "real" information. This is really similar to the second situation, but easier to handle. What you want to do here is use your imagination to get the designee away from the department head so that you can talk privately. Try something like "Gee, Mr. DepartmentHead, you've been extremely helpful and I think from what you've said I have a real good "big picture" of how your department operates. I know you're real busy, so why don't we wrap up and maybe Mr. Designee and I could go find a few of the forms I need copies of and he could show me your present operation." Do whatever it takes, but don't waste time in situations where you don't have control.

4. Good Guy, Bad Guy
   I don't know if you're an avid reader of murder mysteries, but if you are you'll notice that the cops usually interview sources and suspects where one cop plays the good guy and one cop plays the bad guy. If you can find the right person to team up with this can be the absolute optimum way to do user interviewing. The person you choose should be very different from you and should be a "compensating" personality, in other words someone who is strong in areas where you are weak, etc. There are many positive results which can accrue from this approach. First of all, there is a much better chance that at least one of you will develop a close rapport with the user given the differences in personalities. Second, since people think differently, one of you may pick up on things that

the other misses, and so on. Third, rather than really playing good guy, bad guy it is very effective to play smart guy, dumb guy or funny guy, serious guy or whatever seems to be called for given the personalities at hand. If your user seems to be a bit slow, it is helpful for one of you to appear even slower, to set the user at ease. If the user is extremely chatty, one of you can seem equally chatty but the other can continuously pull the conversation back to work.

5. Attitude
   One of the biggest differences between people who make successful interviewers and people who don't, is whether the interviewer goes into the event with the attitude that they are the teacher or they are the pupil. The whole idea of conducting user interviews is to learn how the user does their job, how their business works, what they need to make it work better. The idea is not to show the user how smart you are, how much you know about their job or their business, or how much you know about your job. The interviewer should be a sponge, soaking up as much knowledge and information as possible. Their role is to be the inferior, the pupil, while the user is the superior, the teacher.

6. Technique
   It is very difficult, and takes a lot of time and hard work, to develop a user interviewing technique which works for you. Since every person is different, every person needs their own technique. It must appear and feel natural for their individual personality. My goal, when I conduct a user interview, is to come out of the interviewing process feeling that I know enough to do the person's job. Therefore, the posture I take is to pretend that I am a trainee for the job and that the person I am interviewing will be leaving for vacation as soon as we finish the interviewing process, so I had better have the job down by then. I begin by getting a functional overview of the job, ie. "Let's start with how you get widgets in the first place. Do you make them here at the plant, or do you buy them from outside vendors?" This is then followed by getting in to each step of the job function in more detail. This would logically be the second interview, ie. "Now, the last time we met you said that you order the widgets from an outside vendor. How do you do this? Do you have a blanket order on file with them, do you phone in orders when you're running low, do you send them purchase orders, etc.?" It is critical to this process that you don't have a problem with looking dumb, and asking the same question twenty five times if that's what it takes to understand exactly what is going on. The final step of interviewing, and this might logically be the third interview, is to physically go through the job, ie. watch the way forms are completed, listen to the way phone calls are conducted, listen to the kinds of problems that come up, and so on.

It is very important to know the physical layout and the physical processes that are gone through, as these are frequently just as big a source of inefficiency as the logical processes.

Verifying Your Understanding

Now that all the data has been gathered, you will need to sift it, sort it, and put it into an order that you understand, so that you can start to formulate an idea of how a system might look from the users perspective. You should not be thinking about data base designs and the like at this point. You only want to think about the system through the users eyes. You can cheat to the extent that if you know something is technically impossible, don't mentally design it into the system.

It is a good idea at this point to listen to tapes to refresh your memory, draw diagrams of how a system might flow on your flip charts, and walk thru your thinking with your interviewing partner or other co-workers. Let your mind wander, let your creative juices flow, but think like the user, not like a data processing person.

When you feel confident that you can picture the system in your mind, you will want to set up a meeting with all the users who were in the kick-off meeting and who you have interviewed, and walk through your understanding of the work flow and how the new system will fit into it. What I generally find works for me to accomplish this process is to divide areas or departments into different colors, and using the color coding, draw the entire work flow and new system use out on the flip chart, taping completed pages up in sequence around the walls. This not only provides the user with a very visual feedback, but lets me make changes or re-draw a piece easily if I did not have a correct understanding, and then lets me take the whole thing back to my office to serve as an outline for the final document.

When this walk-thru meeting is complete, everyone in the room should be in agreement that the diarama you have created represents a common understanding of both the way the business operates and the functional requirements of a system which will satisfy the users' needs. Be sure you tape this meeting!

Follow-up

The final step in the process is to follow-up, what should now be a common understanding, with the requirements document and any review or walk-thru meetings which will insure that the document is understood and accepted.

How detailed this document becomes depends on its purpose, your standards, your budget and so on. At a minimum it should include:

1. Table of contents

2. Management Overview

This should be a general narrative of what the system will encompass, what it's approximate cost will be, what the benefits will be and a very brief narrative summary of the functionality of each sub-system.

3. System Description
   On a sub-system by sub-system basis this should outline the functional flow of the system, and name and describe all screens, reports and processes in the system. It may or may not get to the level of a detailed layout of each screen and report, but at a minimum should include the critical information contained on each screen or report, ie. a list of elements.

4. Implementation Plan
   This should contain the proposed scenario of how this system will be implemented, both in terms of hardware and software. It should detail realistic estimates of time and cost. It is basically a "to do" list to get from here to an implemented system.

5. Appendix
   This should include:
   *A Glossary of Terms - there may be many words or terms that are particular to this applicaation or environment. This is especially important if the document will be sent to outside vendors to bid on.
   *Data Dictionary - this should be a complete list, including descriptions and if possible size, type, etc., for every element in the system.

Once this document is published and circulated, you will want to set up at least a brief meeting to address questions and comments. Frequently, if it does not put too much of a strain on peoples time, a full walk-thru of the document is helpful to insure everyone's understanding and concurrence.

Conclusion

The underlying message in this paper is probably already clear to you. In order to effectively understand user requirements you basically need to to do two very important things. First, you must follow all the basic rules of good human communication. Listen carefully, put yourself in the other person's shoes, and feed back what you think you heard. Second, and most important of all, you must forget that you are a data processing professional until the user requirements phase is finished. Otherwise you will confuse your knowledge of resource management, performance considerations, normalized data bases and on and on, with the users functional requirements. Good interviewing!

ASK - Better Than New

Tim Snyder
Boston Scientific Corporation
480 Pleasant Street
Watertown, MA 02172

INTRODUCTION

For those Data Processing professionals who have worked with mainframes and minicomputers and with the HP3000 in particular, the opportunity to work in a department that uses the ASK software is a mixed blessing. ASK's MANMAN system is certainly the most widely installed application software system in use on the HP3000 worldwide, ever. It is becoming one of the most widely used MRP II systems on any hardware (HP3000 and VAX combined).

The reasons for the success of the MANMAN System are not only due to its implementation of the guidelines of MRP II functionality, but also because of its wise use of the particular advantages of the HP3000 itself. The success factors include:

-It is online and the functions are integrated. IMAGE (see Note 1) allows for multiple users concurrently reading and updating multiple databases with a minimal amount of system overhead.

-It is reliable. The HP3000 hardware is very high quality, the operating system (MPE) is mature and reliable, and the DBMS (IMAGE) is extremely reliable.

-It gives predictable and rapid online performance. The operating system and the DBMS combine to give good response time to online users and to degrade in a uniform and predictable manner as it approaches overload.

-It is easy to use. The online HELP functions and the consistent syntax make the packages intructional for new users and fairly intuitive for experienced users.

-It works the same in batch and online. MPE and IMAGE handle batch and online activity through an identical interface and (prior to 6.0) the ASK software functions similarly in batch and in session mode.

-It is extensible without FORTRAN programming. QUIZ provides a powerful and yet easy to use 4GL interface to the data upon which the ASK systems are built while posing no threat to the integrity of the transactions or the data in the databases.

-It is well-documented. The manuals provide user documentation for every command and technical information about the data files accessed/updated by each command.

There are however some weaknesses in the ASK implementation on the HP3000. This paper will address some of those problem areas and most of the topics relate to performance, reliability, or consistency. Some of these weaknesses are totally within the realm of normal technical support responsibilities of the customer; some require minor programming (usually MPE commands and utilities or QUIZ), and others would need basic design or programming changes to address properly. Since much of the success of the MANMAN software is integrally linked to its intelligent use of IMAGE, many of the weaknesses that it contains are related to improper use or inherent limits of IMAGE. For each issue requiring only conventional technical support or light programming, this paper will provide the specifics of how to accomplish those tasks as well as the quantitative performance impact. (see Note 2 for hardware/software configuration)

TECHNICAL SUPPORT ACTIVITIES

Repack Your Datasets

In many cases the database schemas supplied by ASK have the wrong path identified as the primary path in the detail datasets of the IMAGE databases. One can only identify the correct path if one knows the usage pattern of that dataset; and that pattern may vary widely between customers. However, in no case is the correct primary path the one whose path length is always one. Those are often the paths chosen by ASK as primary (SONUM in SOEFIL, REFNUM in ARFIL). The path lengths and other useful information about a database can be viewed by running either DBLOADNG (from the Contributed Software Library from INTEREX) or HOWMESSY (from Robelle). These programs will produce a table (see Figure 1) that will show not only the path length (maximum, average, and standard deviation) but also the inefficieny of the pointers. What this means is the percentage of time that a program must access a different physical block to read the next entry on a chain. Reading a different physical block can take 10 to 100 times as long as reading an entry in the same block. Thus, the performance impact of inefficient pointers on both batch and online performance can be enormous.

To determine which path should be primary, one must find out which path is used most frequently. This can be done using Turbo Profiler from HP (part number 36914A, $3500) if you have converted to Turbo-IMAGE, or by reviewing the programs that access the file. Once the primary path is determined, it may seem useless. Even if you change your primary path (via DBUNLOAD/DBLOAD, ADAGER, DBGENERAL, or DBCHANGE, or other means), the path efficiency will not change significantly over time. This is because the records are added in random order and are placed wherever there is free space.

The only solution is to occassionally repack your detail datasets along the primary path. This can be done via a chained DBUNLOAD/DBLOAD, but for most databases the time involved can be prohibitive. A selective repack for individual datasets is the preferred approach and a number of database utilities (including ADAGER Model I) can do this very rapidly. ADAGER Model I does not even require that the path on which the set is to be repacked be the primary path. You can choose any path.

Repacking does require exclusive access to the database. Be sure to DBSTORE or STORE that database first. This activity should be scheduled for some non-prime time to prevent users from attempting the run applications during the repack. Cryptic error messages advising a call to the "system manager" may result. The pointer efficiency may be reviewed after the repack using DBLOADNG or HOWMESSY again. See Figure 1a for the jobstreams to run DBLOADNG and Figure 1b to run Adager.

How often should sets be repacked? The answer is simple. Often enough but not too often. The schedule for repacking depends on how volatile the data is, how long it takes to do the repack, and how much benefit results. The database utilities DBLOADNG or HOWMESSY can be used to monitor the pointer efficiency. Each user can then decide on how often to do each repack. Some sets (INVFIL in MANDB) may need to be repacked daily while others may only need to be repacked every quarter (INVFIL, SHDFIL).

```
 :JOB DBL6,MANAGER.MANMAN,DATABASE
 PRIORITY = DS; INPRI = 8; TIME = UNLIMITED SECONDS
 JOB NUMBER = #J804
 SUN, JAN 10, 1988,  2:08 PM
 HP3000 / MPE V  G.02.04 (BASE G.02.04).
 :TELLOP BEGIN DBL6 - DATABASE ANALYSIS
 :TELLOP              ANALYZE MANDB.DATABASE.MANMAN,INVFIL
 :RUN DBLOADNG.PUB.UTILITY

 Data Base Loading Analysis -- DBLOADNG  v2.3

 Data Base name:
MANDB.DATABASE.MANMAN
 Password:
ASK
 Mode:
5
 Data set:
INVFIL
 Data set:

 Data Base name:


 END  OF  PROGRAM  :TELLOP  END  DBL6  :EOJ CPU  SEC. = 200.
 ELAPSED MIN. = 9.  SUN, JAN 10, 1988,  2:17 PM
```

Figure 1a. Jobstream to Run DBLOADNG

```
!JOB PACKINV,MANAGER/.MANMAN,DATABASE
!TELLOP BEGIN PACKINV - REPACKS INVFIL IN MANDB
!TELLOP MUST HAVE EXCLUSIVE ACCESS TO MANDB
!ADAGER
MANDB
DETPACK
INVFIL
4
1

5



!TELLOP END PACKINV
!EOJ
```

Figure 1b. Jobstream to Repack INVFIL in MANDB

For example an INVFIL with 8656 records took 13 minutes to repack using option 4 (SUPERCHAINED+) in ADAGER Model I and it reduced the inefficiency from 83% to 4%. The impact on LI,160 was to reduce the number of physical reads for a chain length of 20 from 16 to 2. In another example, SHDFIL with 144303 records took 75 minutes to repack using option 4. The inefficiency went from 66% to 4% on the key HPROD which had an average chain length of 63. Each time RE,830 (Shipping History by Product) is run the average number of physical reads dropped from 42 to 3. A month later INVFIL's inefficiency had risen to 22%.

Override Your Default Database Buffer Specifications

Whenever an IMAGE database is created, one of the parameters indicates how many data buffers should be allocated when the database is opened. The default values for this parameter are the same ones that were appropriate a decade ago when an HP3000 with 512k of memory was considered a large minicomputer. Now that memory is cheaper and therefore more plentiful, one need not be so stingy with data buffers. More buffers can improve database performance. With Turbo-IMAGE, the number can be set as large as desired and the DBMS will only allocate as many as will fit into a data segment. Using IMAGE however, the maximum is limited by the fact that the buffer space and the lock descriptor space share the same extra data segment. Allocating too much space to data buffers could prevent anyone from locking the database.

More significantly, the number of buffers should be set to a single constant number instead of being dependent on the number of users. With a variable number of buffers, IMAGE must interrupt processing everytime the number of users rises or falls by two to expand or shrink the size of the extra data segment. This is simply wasted overhead if your databases are opened hundreds of times each day.

To set the number of buffers in an existing database, sign on as the database creator in the group and account where the database resides (:HELLO MGR.MANMAN,DATABASE). While running DBUTIL (:RUN DBUTIL.PUB.SYS) enter the command

```
>>SET MANDB BUFFSPECS 24(1/120)
```

Separate Your Active Files

Many of the ASK programs and user-written QUIZ reports access more than one file at the same time. One strategy for improving performance is to make sure the files accessed by a single program are on separate disk drives. This can have a significant impact on long transactions (RE,UT commands and long QUIZ reports). With a limited number of disk drives, exactly how to arrange the datasets can become a complex juggling act. ASK does provide some suggestions for 4 and 5 disk drive systems. If you would like to do your own research, the command >LISTF filename;MAP in LISTDIR5.PUB.SYS will show you where each extent of each file is located. Another way to check is to look at the SYSLIST from your full system backups but this only shows the drive of the first extent. For databases and especially if you have recently done a reload, all of the extents will usually be on the same drive. For IMAGE databases the datasets are numbered in the same order as they appear in UT,979 or on the FORM SETS command in QUERY. The two guidelines are: place the masters for each detail dataset on different drives from the detail and from the other masters for that detail and place datasets the are frequently used together on different drives.

Once you have determined which datasets to move to which disk drives, you can choose any number of techniques to accomplish your goal. You can run DBUTIL.PUB.SYS and use the command:

```
>>MOVE MANDB01 TO 3
```

ADAGER Model I will let you move datasets or you can choose device-specific dataset assignment as part of capacity changing or repacking. Be sure to always specify the appropriate disc on subsequent ADAGER operations, otherwise ADAGER might move the dataset to another disk.

When you have moved your datasets, you will also need to check your free space on each disk drive to see if you have overloaded one with too many large datasets.

Restrict Long Transactions To Non-Prime Time

Many of the ASK commands (RE,UT especially) can run for hours on a large database even when there are no other users. For this reason, it is best to limit the users who can run these commands during prime shift by removing them from the list authorized in the password database for those passwords. The same holds true for using QUIZ during prime shift to serially read datasets. Unless a QUIZ report is doing keyed access via the CHOOSE command, the overhead of serially reading moderate to large datasets online during the day using QUIZ can make a noticeable difference in the response time of the machine. Although it can help to run these commands and QUIZ in a lower dispatch queue (either in batch or by forcing them into the DS queue), most of that kind of activity can be scheduled to run on non-prime shift. The impact of executing these commands during prime shift will be felt first and most severely by those doing large data entry transactions (eg Kit a Work Order, Enter a Sales Order, Invoice a Sales Order). If one person is executing a long transaction (over 10 minutes), the response time for the long data entry transactions will double (5 seconds to 10 seconds). If a second person is also doing a long transaction at the same time the response time for data entry will now be triple (15 seconds). It wouldn't take many online reports (QUIZ or RE) or utilities to severly degrade the productivity of the data entry transactions.

Erase Before Using

A number of ASK programs erase entire datasets, and deleting tens of thousands of records from an IMAGE dataset can be very time consuming. One method to improve the performance of those programs is to use a utility to erase the set first. Adager Model I has the SETERASE function and there are other utilities as well that can also erase a dataset much faster than the ASK programs can. The SETERASE function can be completed in less than a minute while it may take hours to delete all the records from MRPPLAN or MPSPLAN while running RE,900 or RE,930.

LIGHT PROGRAMMING

QUIZ Can Be Faster Than FORTRAN

Despite the fact that QUIZ is a Fourth Generation Language (4GL) and a relative newcomer to the world of computer languages compared to FORTRAN, there are many ASK reports that can be written in QUIZ to run an order of magnitude faster than their FORTRAN counterparts. QUIZ is often used in MIS departments that have the ASK software to produce variations on the standard reports. Users often require different selection criteria, different sort sequences, more or less subtotals, more or fewer columns, etc. At the same time QUIZ can also allow the MIS department to get more reporting done in the same amount of time. In this case both the users and the MIS deparment win; better reports in less time.

The two techniques used to accomplish this double-win are the use of QUIZP, the privileged-mode version of QUIZ, combined with the approach of only processing those records that are absolutely necessary.

Every ASK user that has QUIZ also has QUIZP. I have seen QUIZP used as the default version of QUIZ in three ASK user sites every day for years with no problems whatsoever. QUIZP accepts exactly the same syntax as QUIZ and produces exactly the same results only faster. It is faster than QUIZ by an order of magnitude in the simplest situation of serially reading a single file with no LINKS. Whenever LINKS are used, the relative advantage of QUIZP is not as pronounced. This limitation is not as severe as it sounds as shown in the following example.

In OMAR the Backlog Reports (RE,710) can take hours to run if there are tens of thousands of orders in SOEFIL. This can be true even if the backlog is only one order. The reason it takes so long is that the program reads SOEFIL first and for each order examines each record in SODFIL to see if it has been fully shipped. Using QUIZP (see Figure 2), QUF7103 first processes all the records in SODFIL to find the ones on backorder. The first step creates a subfile which is the basis for the subsequent steps which do the linking to other files and which finally produce the report. Other variations of RE,710 could be generated from the same original subfile without adding more than a few minutes to the elapsed time. Figure 3 shows the actual run times when the number of records in SOEFIL were 32,338 and in SODFIL were 77,694. (see Note 2 for hardware configuration)

```
ACCESS SODFIL
DEFINE BO NUMERIC*7 = SODQO - ( SODQS + SODSNI )
SELECT IF BO NE 0
REPORT SUMMARY SONUM SODREQ SODSCH BO SODPRI PRONUM
SET SUBFILE NAME Q7103 TEMP
SET REPORT LIMIT 8000
GO

ACCESS *Q7103 LINK PRONUM TO PRONUM OF PROMAS OPTIONAL
REPORT SUMMARY SONUM SODREQ SODSCH BO SODPRI PRONUM PRODES
SET SUBFILE NAME Q7103A
GO

ACCESS *Q7103A LINK PRONUM TO ITNO OF IM OPTIONAL
REPORT SUMMARY SONUM SODREQ SODSCH BO SODPRI PRONUM PRODES &
               QOH
SET SUBFILE NAME Q7103B
GO

ACCESS *Q7103B LINK SONUM TO SONUM OF SOEFIL OPTIONAL
SORT ON PRONUM ON SODREQ
DEFINE DSCHDATE DATE MMDDYY=DATE  (SODSCH  + 26235 )
DEFINE DSODTOT  NUMERIC*9  =BO*SODPRI
DEFINE DSODPRI NUMERIC*9   =SODPRI
DEFINE HOLD STRING*1= "*" IF SOEHLD NE 0 ELSE &
  " "

REPORT  TAB 1 HOLD TAB 2 PRONUM TAB 17 SHPNO TAB 28 PRODES &
  TAB 51 SOES1 TAB 81 DSCHDATE TAB 90 SONUM TAB 99 BO &
  TAB 108 DSODPRI TAB 120 DSODTOT
FOOTING  AT PRONUM  SKIP 2  &
    TAB  43 " SUBTOTAL   " PRONUM  &
    TAB 67 "QTY-ON-HAND" QOH  &
    TAB  99 BO  SUBTOTAL      &
    TAB  120 DSODTOT  SUBTOTAL  SKIP 2

FINAL   FOOTING    &
     TAB   93  "FINAL TOTAL  "   &
     TAB   120  DSODTOT   SUBTOTAL

PAGE HEADING TAB 45 "COMPANY NAME" SKIP 1 TAB 2 SYSDATE &
 TAB 15 SYSTIME TAB 40 "DAILY INVENTORY TO BACKLOG" &
 TAB 105 "PAGE" SYSPAGE SKIP 1 TAB 1 "REPORT QUF7103" &
 TAB 42 " SORTED BY PRODUCT NUMBER" SKIP 1
SET REPORT DEVICE PRINTER
SET REPORT NAME QUF7103
SET REPORT LIMIT 8000
GO
```

FIGURE 2. QUIZ Report to imitate RE,710,5 only faster

```
:JOB TEST7103,SNYDER.MANMTI,SNYDER
PRIORITY = DS; INPRI = 8; TIME = UNLIMITED SECONDS
JOB NUMBER = #J802
SUN, JAN 10, 1988, 12:50 PM
HP3000 / MPE V  G.02.04 (BASE G.02.04).
:QUIZP 01,01
Q U I Z   (5.01.E1)F     LEVEL 50   06014
Copyright 1985 COGNOS INCORPORATED
> USE QUF7103.QUIZMTI
> GO
*W* Attempting to open primary file in privileged mode.
*W* Primary file opened with privileged mode.

Records selected:  5226
Records written:   5226

Records selected:  5226
Records written:   5226

Records selected:  5226
Records written:   5226

Records selected:  5226
Records sorted:    5226
Lines printed:     6639
Pages printed:     139

> SHOW ACTIVITY

Elapsed time (minutes):  11
CPU time (seconds):      342
Files opened:            7
Records read:            108803
Records selected:        20904
Records sorted:          5226
Records written:         15678
Records reported:        20904
Lines printed:           6639
Pages printed:           139
Reports produced:        4

> EXIT

END OF PROGRAM
:EOJ
CPU SEC. = 356.ELAPSED MIN. = 12.SUN, JAN 10, 1988,1:01 PM
```

FIGURE 3.   STDLIST from Running QUIZ Report

Use the CASE Contruct In QUIZ

For QUIZ (5.01) reports, one suggestion is to use the CASE construct instead of IF...ELSE in DEFINEd temporary variables. Since the options of the CASE statement (equal to one value or between two values) are much simpler than the Boolean nightmares that can be constructed with IF...ELSE, the results can be evaluated using a simpler and faster algorithm. The following is a CASE (!) study of exactly how much faster.

The report (QUF102) in Figure 4. refers to a usefile (PTLEVEL4) in Figure 5 that defines a four-level hierarchy for grouping products by product type in Ask's Order Processing System - OMAR. The original usefile contained eight DEFINE statements which contained a total of 117 IF clauses. The new usefile has the same DEFINE statements with 117 WHEN clauses.

The report sorts by product number within product type within each of the four levels of the hierarchy to create a listing of all products with HEADINGs, FOOTINGs and COUNTs at each of the four levels. The file contains 5856 records of which 3154 are selected, sorted and reported. The IF...ELSE syntax required 129 CPU seconds over 35 minutes whereas the CASE syntax required 69 CPU seconds over 10 minutes. (See Note 2 for Configuration)

The results of this test were sufficiently significant (the faster syntax required only 53% of the CPU time) that the guidelines for QUIZ reporting now specify the use of the CASE syntax whenever possible in this MIS department.

Since this particular usefile is referenced by forty reports, the benefits of this improvement will be immediately incorporated into a large portion of the regular reports for this company. Perhaps this will delay the need for that hardware upgrade by a few weeks.

```
ACCESS PROMAS LINK PROTYP TO PTCODE OF PTMAS
DEFINE DPROTYP NUMERIC*10 = PTCODE
USE PTLEVEL4.QUIZMTI.MANMTI
SORT ON LEVEL1 ON LEVEL2 ON LEVEL3 ON LEVEL4 ON PROTYP &
     ON PRONUM
PAGE HEADING TAB 1 "DATE: " SYSDATE &
             TAB 20 "TIME: " SYSTIME &
             TAB 120 "PAGE: " SYSPAGE &
             SKIP &
             TAB 1 "PROGRAM: QUF102" &
             TAB 50 "PRODUCT LISTING BY PRODUCT TYPE" &
             SKIP &
        TAB 50 "========================================" &
             SKIP 2 &
             TAB 1 "DESCRIPTION" &
             TAB 35 "PART NUMBER" &
             TAB 60 "PROD TYPE" &
             TAB 72 "UOM" &
             SKIP 2
REPORT       TAB 1 PRODES &
             TAB 35 PRONUM &
             TAB 60 PROTYP &
             TAB 72 PROUOM &
             SKIP
HEADING AT LEVEL1 TAB 1 HEAD1
HEADING AT LEVEL2 TAB 3 HEAD2
HEADING AT LEVEL3 TAB 5 HEAD3
HEADING AT LEVEL4 TAB 7 HEAD4
HEADING AT PROTYP TAB 9 PTDESC
FOOTING AT PROTYP &
    TAB 9 PTDESC "COUNT=" &
    TAB 40 COUNT SKIP 2
FOOTING AT LEVEL4 &
    TAB 7 HEAD4 "COUNT=" &
    TAB 40 COUNT SKIP 3
FOOTING AT LEVEL3 &
    TAB 5 HEAD3 "COUNT=" &
    TAB 40 COUNT SKIP 4
FOOTING AT LEVEL2 &
    TAB 3 HEAD2 "COUNT=" &
    TAB 40 COUNT SKIP 5
FOOTING AT LEVEL1 &
    TAB 1 HEAD1 "COUNT=" &
    TAB 40 COUNT SKIP 6
FINAL FOOTING SKIP 5 TAB 1 "TOTAL COUNT=" TAB 20 COUNT
SET REPORT LIMIT 10000 DEVICE PRINTER PRIORITY 5 COPIES 1
SET REPORT NAME QUF102
GO
```

FIGURE 4.   QUIZ Report that Refers to Usefile

```
DEFINE LEVEL1 STRING*1=  CASE OF DPROTYP &
       WHEN 1000 TO 24999 THEN "1" &
       WHEN 25000 TO 26999 THEN "2" &
       WHEN 27000 TO 29998 THEN "3" &
           DEFAULT "4"
DEFINE HEAD1 STRING*20=  CASE OF LEVEL1 &
       WHEN "1" THEN "TOTAL TRADE" &
       WHEN "2" THEN "TOTAL OEM" &
       WHEN "3" THEN "TOTAL INTERCOMPANY" &
           DEFAULT "MISC"
DEFINE LEVEL2 STRING*1=  CASE OF DPROTYP &
       WHEN 1 TO 11999 THEN "1" &
       WHEN 12000 TO 21999 THEN "2" &
       WHEN 22000 TO 24999 THEN "3" &
           DEFAULT "7"
DEFINE HEAD2 STRING*15=  CASE OF LEVEL2 &
       WHEN "1" THEN "THERAPEUTIC" &
       WHEN "2" THEN "DIAGNOSTIC" &
       WHEN "3" THEN "SURGERY" &
           DEFAULT "MISC"
DEFINE LEVEL3 STRING*2=  CASE OF DPROTYP &
       WHEN 1000 TO 2999 THEN " 1" &
       WHEN 3000 TO 4999 THEN " 2" &
       WHEN 5000 TO 11999 THEN " 3" &
       .  .  .  .
           DEFAULT "12"
DEFINE HEAD3 STRING*20=  CASE OF LEVEL3 &
       WHEN " 1" THEN "VASCULAR" &
       WHEN " 2" THEN "DRAINAGE" &
       WHEN " 3" THEN "URORADIOLOGY" &
       .  .  .  .

           DEFAULT "MISC"
DEFINE LEVEL4 STRING*2=  CASE OF DPROTYP &
       WHEN 1000 TO 1099 THEN " 1" &
       WHEN 1100 TO 1199 THEN " 2" &
       WHEN 1200 TO 1299 THEN " 3" &
       WHEN 1300 TO 1399 THEN " 4" &
       .  .  .  .
           DEFAULT "40"
DEFINE HEAD4 STRING*20=  CASE OF LEVEL4 &
       WHEN " 1" THEN "STEERABLES" &
       WHEN " 2" THEN "OCCLUSION BALLOONS" &
       WHEN " 3" THEN "DILATATION CATH/PEMT" &
       WHEN " 4" THEN "BLUE MAX" &
       .  .  .  .
           DEFAULT "MISC"
```

FIGURE 5.   USEFILE with CASE statements (abbreviated)

Use QUIZ To Switch Your Transaction Logs Before They Fill Up

One of the most unpleasant surprises is to have your MANMAN/MFG system shut itself down during the day without warning. One easy way to do this is to fill up the Transaction Log file. When this occurs, all users must log off (or be logged off) from the system; and the user who shut it down must start it up again. If you are lucky, no transactions have been lost and the only long-term effect is a few dozen (or hundred) mildly aggravated users.

It would be nice if ASK provided a utility which would decide based on some COMIN variable setting whether or not to switch logfiles when doing the nightly UT,990/UT,999. One certainly does not want to create a new logfile every day. If the second shift operator is dependable, s/he can be assigned the task of daily monitoring the need to switch logfiles.

There is a way to automate the task without a single line of FORTRAN code. QUIZ and a series of jobstreams can be used to check the EOF on the TR file; and if it exceeds a certain value, it can choose between streaming a job to switch or one not to switch the logfiles. These jobs can be part of your nightly schedule of unattended activities but should be done before any other jobs or sessions are permitted after the backups.

Figure 6a. shows the dictionary that defines the format of the LISTF,2 for QUIZ. The jobstream in Figure 6b. will determine if the most recent Transaction Log File has exceeded 20,000 records (capacity set to 30,000). If it has, then the subsequent job STOPMANW will switch to a new Log File; whereas STOPMAND will do the shut down/start up but will not switch logfiles.

```
SCHEMA "     LISTF FORMAT                   "

FILE LISTFILE         TYPE MPE                  &
  OPEN LISTFILE

ELEMENT FNAME            X(008) HEADING "FILE NAME  "
ELEMENT EOF     9(008)         HEADING "END-OF-FILE"
ELEMENT FILLER23         X(023)
ELEMENT FILLER41         X(041)

RECORD LISTFILE
  ITEM FNAME                      CHARACTER
  ITEM FILLER23                   CHARACTER
  ITEM EOF                        DECIMAL
  ITEM FILLER41                   CHARACTER

BUILD SCHEMA
```

FIGURE 6a.   QUIZ Dictionary Definition of LISTF,2 file

```
!JOB STOPMAN,BATCHM/BATCHM.MANMAN;inpri=13
!TELLOP BEGIN STOPMAN-GENERATE JOB TO SHUT/RESTART MANMAN
!PURGE LISTFILE
!LISTF TR@.DATABASE.MANMAN,2;LISTFILE
!SAVE LISTFILE
!EDITOR
T LISTFILE
CQ 1/1,"",ALL
DQ 1/5
SET FIXED
SET RIGHT=80
SET LENGTH=80
K
EXIT
!FILE QUIZLIST=$NULL
!FILE QSCHEMAC=LISTFQ
!QUIZ
ACCESS LISTFILE
SORT ON FNAME D
SET SUBFILE NAME LTEMP
REPORT SUMMARY ALL
GO
ACCESS *LTEMP
SET REPORT LIMIT 1
SET REPORT DEVICE DISC NAME STOPMANT
SET NOHEAD NOWAIT NOBLANKS
DEFINE MAXEOF STRING*17 = "STOPMANW.JOB" IF EOF > 20000 &
                      ELSE "STOPMAND.JOB"
INITIAL HEADING &
  TAB 1 "!JOB STOPMANT,BATCHM/BATCHM.MANMAN;INPRI=13" SKIP &
  TAB 1 "!TELLOP BEGIN STOPMANT"
REPORT TAB 1 "!STREAM" TAB 13 MAXEOF
FINAL FOOTING  TAB 1 "!RUN PAUSE" SKIP TAB 1 "30" &
          SKIP TAB 1 "!TELLOP END STOPMANT" &
          SKIP TAB 1 "!SET STDLIST=DELETE" &
          SKIP TAB 1 "!EOJ"
GO
EXIT
!STREAM STOPMANT
!TELLOP END STOPMAN
!SET STDLIST=DELETE
!EOJ
```

Figure 6b.   Jobstream to Switch Logfile when Appropriate

Why Isn't The COMIN File In Your Dictionary?

QUIZ can be used in most of the ASK systems to access all the data that the system uses. There are a few files in some of the systems that are not included in the Dictionary supplied by ASK and therefore, cannot be accessed with QUIZ. The dictionary can be modified to add new files and new elements. One important source of data is the COMIN file in MANMAN/MFG and in OMAR. The number of days/months usage and the fiscal period are two pieces of information which are necessary for producing useful QUIZ reports and which reside in the COMIN files.

Prior to version 5.01, the method that QUIZ used to open MPE files (ie COMIN) made it difficult to allow QUIZ to access these files while the ASK system was updating them. QUIZ would lock out the ASK programs which was a disaster for the users. The newer version of QUIZ, however, does not lock out ASK users nor is it locked out by ASK users.

Figure 7a. shows the entries that need to be added to the QUIZ Dictionary. After using the editor to modify the source (QSCHEMA.QUIZ.MANMAN), the Cognos utility QDD.CURRENT.COGNOS must be run to recompile the Dictionary. The recompilation may take a while and will make the compiled dictionary (QSCHEMAC.QUIZ.MANMAN) inaccessible for some time. Plan this task for a time when no one is using QUIZ. Figure 7b. shows a simple QUIZ report to access the number of months usage from the COMIN file and to calculate the average monthly usage.

```
FILE MCOMIN TYPE MPE OPEN COMIN/LOCK.DATABASE

ELEMENT COMINEL  9(005)
ELEMENT COMINELX X(002)

RECORD MCOMIN
  ITEM COMINEL     INTEGER SIGNED SIZE 2
  REDEFINED BY
    ITEM COMINELX  CHARACTER SIZE 2
  END
```

FIGURE 7a.   QUIZ Source Dictionary Code for COMIN File

```
ACCESS IM LINK TO RECORD 9 OF MCOMIN
CHOOSE ITNO PARM
DEFINE AVGUSG NUMERIC*8 = PTDUNM / COMINEL
REPORT ITNO AVGUSG
GO
```

FIGURE 7b. QUIZ Report Using the COMIN file

What's New Pussycat?

One type of question to which users frequently require an answer is "Who are the new customers?" or "What parts/products are new this year?" The ASK systems track most of the activity files by date (sales orders, purchase order, work orders) but none of the master files indicate when the record was added. Perhaps someday ASK will add the logic to its AD,nnn programs to record the two-byte ASK formatted date in IM,PROMAS,CUSFIL,BILMAS,etc.; but one need not wait. Again QUIZ can provide a work-around.

On a specified schedule (weekly, monthly, yearly) a simple QUIZ report can be run to copy the unique keys from the file (eg PRONUM from PROMAS) into a permanent subfile (see Figure 8a.). The subfile can them be renamed with an appropriate suffix (eg :RENAME PROMAS,PR880101 for January 1, 1988). The subfile can then be used to generate a report at a later date of all those products on PROMAS which were not in PROMAS on 1/1/88 (see Figure 8b.). This approach assumes that products are never deleted, but a slight variation could address that situation as well.

```
ACCESS PROMAS
REPORT SUMMARY PRONUM
SET SUBFILE NAME PROMAS KEEP
SET REPORT LIMIT 9000
GO
```

FIGURE 8a. QUIZ Report to Create Snapshot Subfile of PROMAS

```
ACCESS *PROMAS
REPORT SUMMARY ALL
SET SUBFILE NAME PROTEMP
SET REPORT LIMIT 20000
GO
ACCESS PROMAS
DEFINE PROCOUNT NUMERIC*2 = 1
DEFINE COUNTDATE DATE     = SYSDATE
SET REPORT LIMIT 20000
REPORT SUMMARY PRONUM PROCOUNT COUNTDATE
SET SUBFILE NAME PROTEMP APPEND
GO
ACCESS *PROTEMP
SORT ON PRONUM
REPORT SUMMARY PRONUM PROCOUNT SUBTOTAL COUNTDATE
SET SUBFILE AT PRONUM NAME PROTEMP2
GO
ACCESS *PROTEMP2 LINK TO PROMAS
DEFINE SINCE YYMMDD = PARM
SELECT IF PROCOUNT = 1
PAGE HEADING TAB 1 "DATE: " SYSDATE &
             TAB 20 "TIME: " SYSTIME &
             TAB 120 "PAGE: " SYSPAGE &
             SKIP &
             TAB 1 "PROGRAM: QUN002 " &
             TAB 55 "NEW PRODUCTS ON OMAR               " &
             SKIP &
             TAB 55 "SINCE                 " &
             TAB 78 SINCE &
             SKIP 2 &
             TAB 1 "PRODUCT" SKIP &
             TAB 1"NUMBER" &
             TAB 18 "DESCRIPTION" SKIP 2
REPORT TAB 1 PRONUM &
       TAB 18 PRODES
SET REPORT DEVICE PRINTER NAME QUN002 PRIORITY 5
GO
```

FIGURE 8b. QUIZ Report Showing New Products in PROMAS

FLAWS WE HAVE LEARNED TO LIVE WITH

Why Not IMAGE?

Many of the newer ASK systems have all the files including the COMIN file defined as part of the database. Why haven't the older systems been brought up to date to include all the data in the IMAGE database? This dichotomy between IMAGE and non-IMAGE files complicates backup, recovery, support, programming, QUIZ access.

As a corollary problem, dealing with data in the non-IMAGE COMIN files can be very difficult especially if the format is a double word integer split into two COMIN variables. These parameter files should be easier to work with.

Why Not Transaction Logging?

Since much of the data for the ASK systems (MFG,OMAR,AP) exists outside the IMAGE database, the possibility of using IMAGE logging (Roll-Forward, ILR, or Roll-Back) is very remote. The options for ILR and Roll-Back recovery are fairly new and there is a performance price to be paid for any of these techniques; but ASK could allow users to make their own choices about the trade-offs between reliability and performance. Even with many of the files outside IMAGE, the programs could at least include DBBEGIN/DBEND blocks to bracket logical transactions. If a user is not using logging, the DBBEGIN/DBEND are ignored by IMAGE and take a miniscule amount of CPU time compared to the whole transaction.

Why Integer Keys?

There are a number of keys for IMAGE datasets in the ASK system which represent ticking time bombs. Most of the keys in the databases are character data elements but several are integers (PTCODE in OMAR, CHKNO in PAYDB). These keys can cause severe performance problems if the values of the keys begin to generate duplicate values for a remainder when divided by the capacity of the dataset. These keys could be converted to right-justified zero-filled character strings to preserve the sort sequence while avoiding the hashing problem that integer keys can often cause.

Data Element Numbers, Names, Item Lists, and *

The ASK databases violate one of the most basic guidelines for designing normalized databases. The same data element is given different names in each dataset. This practice makes the structure of the database seem far more complex than it is and makes programming and QUIZ reporting far more difficult. Even more troublesome is the practice of using the same element to mean two different things. In ARFIL the element SONUM can mean the original Sales Order Number or it might mean the referenced invoice number for a credit memo. If it is a duck, call it a duck; and if it is not a duck, don't call it a duck.

When referencing an item list in a database call (DBGET,etc), the ASK programs use item numbers. This practice may seem efficient but it is very rigid and is not much more efficient than using the special lists (@ for all elements, or * for the same list as previously used). Item number lists could also be used without hard coding the item numbers into the programs. The IMAGE Intrinsic DBINFO will convert item names to numbers and remove the need to predefine all elements in a database (DUM001,etc.).

Command Logging Could Be More Useful

In version 5.0 ASK provided the option of logging the commands being used to a disk file. This was a wonderful idea but it was incompletely implemented. The records did not properly record the CPU time consumed by sub-processes (RE,UT and some TR commands) and the file was not defined in the POWERHOUSE dictionary. Since the information is incomplete and since access is limited to a single report, this potentially valuable tool is almost useless.

Database Locking: Sometimes too Much, Sometimes too Little

All the systems do all database locking unconditionally. Why couldn't the choice be left to the users via a COMIN variable as to what kind of locking will be done? Some of the utilities lock after the open and unlock before the close of the database when the lock could be issued around each update or could be restricted to a single file instead of the whole database. In some cases less global locking would be more compatible with other users of the database.

Many reports read a dataset, create a work file with record numbers from the dataset, sort the workfile, reread the workfile and try to reaccess the records in the dataset by record number. This occassionally results in IMAGE errors when the record has been deleted while the report was in progress. If this two-step reporting is going to be used, why doesn't the program lock the set until it is complete?

CONCLUSION

The ASK Software is superior in functionality and performance on the HP3000, but there is always room for improvement. Each ASK user can add to the success of his/her environment by optimizing the performance of his/her own activities. ASK will continue to improve their software and may include some of these ideas in the future.

NOTE 1. Throughout this paper IMAGE refers to either IMAGE or Turbo-IMAGE interchangeably unless otherwise noted.

NOTE 2. The hardware configuration for most of the statistics was an HP3000/68 with 5 Megabytes of memory, 5 7933XP disc drives, two IMBs and three high speed GICs with Memory Caching turned off running UB delta 4 MIT (G.02.04) of MPE and Version 5 of ASK Software.

CONTINGENCY PLANNING -- THE AUDIT PROCESS

Leslie A. Virgilio
OFF-SITE, Inc.
32 Ellicott Street
Batavia, New York 14020

Disaster Recovery is the ability to continue your information processing when your facilities for doing so are unavailable. Situations requiring recovery can be natural disasters, industrial accidents, human relations, and hardware failure. None of these are recoverable without a Contingency Plan. The Disaster Recovery Strategy protects against the improbable. Contingency Planning prepares you for the inevitable.

Companies should insure their computer hardware for the replacement costs involved. Along with this policy they may also have an "ability-to-operate" clause which guarantees them some income should they have to close business due to data processing failures. Most insurance policies protect against immediate financial loss due to disaster. Other losses such as client base, reliability of service, cash flow, payroll calculations, and reporting capabilities are not recoverable on insurance policies.

Auditors are now requesting that part of their clients corporate profile be a Contingency Plan and Disaster Recovery Strategy. Some organizations may even be pressured by government agencies to prepare such a plan. Yet, when companies are asked whether or not they have a Contingency Plan for their Data Processing needs, the answer is often, "Yes, we back up our system and store our tapes at a remote location." The problem with this answer is what do you do with those tapes if your machine is unavailable to use due to a disaster situation? Several options are available.

One option is a Private Backup Site. These sites are owned by the business involved. To be of full benefit in the case of a disaster, this site should be in a different location than the original. There are two types of private backup sites: "cold" and "hot". A "cold" site is a fully equipped computer facility, without the computer. Only electrical power, air conditioning, and telecommunications equipment exist. When disaster strikes, the computer and required peripherals must be obtained, installed, and tested. Although relatively low in cost, the "cold" site has the disadvantage of a lengthy implementation. A "hot" site is a fully equipped computer facility with an identical or very similar computer system to the original, already installed. Obviously, the most desirable system from an operations standpoint, this alternative is extremely expensive. Another drawback to this alternative is the easy justification for using the system/facility for other uses. This eliminates the 100% availability for disaster recovery.

A second option is a Mutual Backup or Reciprocal Agreement. A Mutual Backup Agreement can be between two businesses, or between two different computer sites within the same business, with similar system configurations. They agree to back up one another should a disaster occur. The businesses are usually located near each other. To eliminate competition, the companies are usually in different industries. Although there is little or no cost to the agreement, there are several drawbacks. Few corporations will allow a second or third level manager to form a binding contract on a handshake. There are very few sites with sufficient excess capacity to operate a second business without curtailing their own operations. Will your CEO allow his business to fall behind to allow another company to use his hardware? Further, the most critical phase of Disaster Planning is testing. This is the step most commonly omitted from mutual agreements. For these agreements to really work, companies would have to have far more computer capacity than their businesses require.

A third option is the "Cold" Backup Site, which is similar to the privately owned cold backup site. It is an "empty shell" facility owned and operated by a company in the business of data disaster recovery. Cold Sites bring up the term "Allowable Downtime". How long can you be without a computer? How long will it take to "warm up" a cold site? Is it conceivable that a hardware configuration sufficient to support operations can be delivered, installed, and made operational within a sufficient amount of time to keep the company running efficiently? An open purchase order with a vendor for delivery of a complete configuration only guarantees purchase of the equipment, not that it will be delivered when needed.

A fourth option is a Remote Site. This type of facility depends on telecommunications. Dialing into a computer system can create problems for the users. The number of external forces working against the ability to exchange information are staggering: weather, traffic accidents, power failures, and load switching just to mention a few. Also, you must be sufficiently supplied with modems, at reasonable working speeds, and terminals to be able to use the Remote Site.

A fifth option is a Mobile Site. The Mobile Site again brings up the term "Allowable Downtime". Alot of computing power can be put in the back of a truck, but how soon can you get it where it is needed? And at what cost?

Another option is the Hot Backup Site. This situation is probably the most acceptable solution to disaster recovery. It is a fully equipped computer facility, owned and operated by a company in the business of disaster recovery. Although there can be competition for its use, disaster recovery companies can often compensate by having multiple CPU's and/or multiple hot site locations. Often, the disaster recovery facility can also accommodate users by having terminals/workstations for people to use.

The choice of where to recover must meet the needs of the company. Hewlett-Packard has provided us with computing hardware compatability unsurpassed in the industry; an interchangeable operating system. For the most part, any software will run on any machine just by using two MPE commands: RESTORE and RUN. Therefore, any of the above options will work.

Assuming you have solved the hardware problem, what about your users? A workable Disaster Recovery Strategy and Contingency Plan requires not only hardware to continue operations, but also a transferable set of software and users.

What then should the approach to Auditing and Contingency Planning be? Ask yourself "What's wrong with the existing methods of preparing for a disaster?" The answer is simple. We write up a set of procedures, document systems, define requirements, ignore the users, put it on a shelf and never look at it again. For a Contingency Plan to work, the document must become a useful tool; something that will be a part of our daily operations and decision making. If it is used daily, it will be updated. Having current information is the only way any Contingency Plan can work.

Basic DP Audits are offered by most public auditing firms as part of the annual Financial Audit. These audits cover procedures and data flow, usually tracking specific portions of information in order to understand their source. This information should be incorporated into the Contingency Plan. However, a complete plan must also include the mechanics of operation. It must be developed by individuals who know and understand the computer systems being utilized as well as the information processing needs and methods of the organization. The only way to truly accomplish this is through the Data Processing Audit. This Audit includes complete definition of the Data Processing System, both manual and computerized. Identification of each application and the subsets of these applications are also defined. Within the subsets, key personnel, special requirements such as source documents and output forms, as well as the relationship between applications, are revealed.

It is not good enough for the Contingency Plan to tell us only what to do when a system fails. It must guide us when the individual component parts of the system go astray. These component part failures come in a variety of ways. The most common in any organization is key user vacation time and extended sick leave. Moreover, from a computerized standpoint, if data becomes corrupted or application software fails, which related applications will no longer function? As mentioned earlier, we are dealing with the inevitable. People will change jobs. Hardware systems will fail. Processing will need to be stopped. By understanding the interrelationships and needs of the data processing function, it becomes possible to prepare for these inevitable situations.

Contingency Planning cannot be restricted to the computerized flow of information. It must include those manual procedures required to supply the flow and support those which are computer dependent. In the event of a complete systems disaster, such as fire, it is also necessary for the Contingency Plan to identify which applications are critical to daily business; which applications need to be put into place first. Fortunately, the Data Processing Audit identifies the applications most critical to the organization and, of these, what other applications are dependent and which are related. We now have the ability to put into place portions of the overall system versus restarting of the entire process.

There are several factors that should be considered when doing a Data Processing Audit. These include: hardware resources utilization & requirements; primary & secondary systems support equipment; vendors; forms; software applications; personnel -- duties, responsibilities, back-up and schedules; emergency calling; subordinates; risk analysis -- resources, environment, personnel and software; critical processing timetable; allowable downtime. Let's look at each of these critical areas independently.

HARDWARE RESOURCES UTILIZATION AND REQUIREMENTS

When evaluating hardware resources for disaster recovery planning, we need to know what the minimum requirements needed to be able to function in a contingency mode are. In order to determine that, we need to know current hardware configurations, including: operating system MIT; computer series; megabytes of main memory; number of printers and LPM speeds; number of modems and baud rates required; number of Mux. channels; number of INP boards; number of modem links; number of tape drives and BPI speeds; disc space utilized; number of terminals needed and if any special terminals are needed for added memory or graphics capabilities; special equipment such as bar code readers and optical scanners.

PRIMARY & SECONDARY SYSTEMS SUPPORT EQUIPMENT

For each site location, we need to know about the environment. What type of power control equipment do we have? What type of environment control equipment? Who are the vendors, the contacts? What company provides our power source? Do we have fire protection? If yes, what type? Halon, water sprinklers? What type of structure is the computer room? Are there fire walls? If there is a fire outside the computer room, how much time do we have before the computer room catches fire as well? All this information is vital to be able to rebuild the type of facility you currently have and/or to be able to salvage what currently exists. These factors also determine the disaster risks and survival abilities.

VENDORS

Computer supplies and other equipment needed to run your systems may also be inaccessible in a disaster situation. A list of vendors with purchase order numbers, inventory lists, and other information is crucial to facilitate replacements. Information about software vendors is needed as well. Does the company provide telesupport and/or site support? Who is the primary contact? Has the vendor given approval for the use of their software on an alternate machine? You want this information easily accessible.

FORMS

Identification of forms must also be done. What are the forms used in the applications? Who is the vendor? What is the order unit of measure? What is the monthly usage? What is the order lead time? Where are the forms stored? What applications use the form and what is the consumption by the application? Forms identification not only applies to preprinted output forms. It should include manually prepared source documents that are needed.

SOFTWARE APPLICATIONS

Software applications can have one of three characteristics. They can be dependent, independent or associated. Independent applications are those which will function as self-contained units regardless of the existance of any other applications. Dependent applications are those which require interaction between two different applications for the purpose of decision making. Associated applications are those which utilize portions of other systems in a passive manner. For each application, dependent and associated applications must be identified. Each application user must be identified as well as their duties and responsibilities. Back up personnel must be assigned to each user. Who are the in-house technicians? Is there a vendor software engineer? If yes, who is it? Where can he/she be reached and at what hours? What type of application are we running? Finance, order entry, etc. What languages is the application written in? What types of files does it require? If it's a purchased application, have any of the programs been customized? How many terminals are needed to run the application? How many megabytes of disc spaces? How many people? What is the Allowable Downtime? Which computer installation is used for this applications processing? For each subset of the application, the transaction volume and required transaction turnaround time must be defined. The critical processing times of each subset must also be defined, as well as the duration. We also need to know what special equipment, whether computer or non-computer, is needed to run each application successfully. For instance, when running accounts payable, the checks may need to be printed on a special printer, bursted by a burster, folded and sealed by a folding machine and then stamped by a postage meter. We need to define if the equipment is critical or merely useful to the processing. A most critical question...has the vendor approved use of the software at an alternate site in a disaster situation? Does the application require special forms? If yes, are they critical or just useful?

PERSONNEL -- DUTIES, RESPONSIBILITIES, BACK-UP, SCHEDULES

Who are the key people in the information flow? They know who they are, but how many of us have assumed responsibilities, out of necessity, that our direct supervisors are unaware of? The relationships between people and their work are similar to the relationships between hardware and software and between software applications. Knowing how the people relate to the work performed is just as important as knowing how the hardware and software relate to each other. This is the mechanics of operation, the manual process required to support the electrical process.

EMERGENCY CALLING

An Emergency Calling List is a list of all key people, in call-priority order. Supervisory personnel should be given the highest call priority since they should be the first to be notified in the case of a disaster.

SUBORDINATES

All key personnel need to be listed by their supervisor. In the case of a disaster, each supervisor needs to know who they need to contact and what the appropriate phone numbers are.

RISK ANALYSIS -- RESOURCES, ENVIRONMENT, PERSONNEL, SOFTWARE

This area is very critical. There are several ways of reducing the risk of a disaster from protection of computer data to protection of data center operations; from protection of vital user records to insurance. A successful risk analysis will identify areas that are lacking. Areas that, if not taken care of, could be partially responsible for a data processing disaster.

CRITICAL PROCESSING TIMETABLE

Critical processing periods are designated for each system and/or subsystem. This information indicates how long an application can be unavailable before it is needed again. It will also indicate how long the application needs to run to successfully complete. Since this information varies from day to day, it is more or less represented in calendar form. Special processing periods (end of month, quarter, etc.) are also specified. All this is taken into consideration when making judgments about data recovery.

ALLOWABLE DOWNTIME

How long can the company survive without a computer system? The allowable downtime can be dependent on the day of the week, day of the month, and/or time of day. Typically, allowable downtime is the length of time between the running of critical applications.

With all this related information pulled together, the Contingency Plan emerges. But more than just a Contingency Plan, you also have an Audit Report that defines the mechanics of operations, the relationships of applications, the key users and their schedules, and the special requirements required to support the electronic flow of information. A document that, because it is used on a daily basis, will be kept current.

Preparation for the inevitable must begin with foresight. If we are to protect our business, and ourselves, against catastrophe and limit disaster, our data must be sound, our plan current, and our resources assured. The steps that must be taken are: making provisions for the replacement costs of data processing equipment, including the facility itself; auditing and documenting the data processing situation; updating the document as required; selecting a recovery site and binding it with a contract; and testing, at least once a year, to make sure the plan works. Anything less, and the prepartion for the inevitable could turn to disaster.

Fortran 66 to 77 Conversion: Problem Considerations in an Integrated Environment

Brant L. Kelly
Bradmark Computer Systems
4265 San Felipe
Suite 820
Houston, Texas 77027

Introduction

With the introduction of the Precision Architecture line of computers, HP has made its Fortran 66 compiler obsolete. Along with the introduction of the Fortran 77 compiler, HP provided some tools to help the programmer in converting Fortran 66 source to Fortran 77 source. After converting a system written in Fortran 66 to Fortran 77, I gleaned some knowledge that is not clear in the documentation. The system that I converted is a group of fortran subroutines that calls system intrinsics. The steps outlined in the migration manual for converting Fortran 66 source are clear enough, so this paper is not a "how to" guide. The purpose of this paper is to share my experience in a conversion of interacting programs in a system. The conversion facility from HP is a good tool if the programmer has some previous knowledge on its use. This paper covers internal representation of Fortran 77 data types with the interest of helping the programmer in calling external routines that do not use the same default data sizes as Fortran 77.

I. After the Migration Aid: What Now?

You spent hours of reading and experimenting to understand just what the Migration Aid will do. After taking the plunge, you now have a file of fresh FORTRAN 77/V source code just itching to be compiled. Will it work? If it is a stand-alone program that does not call other programs or system intrinsics, then it will work (probably). If your program is a routine that interfaces with other programs on your HP3000, then there is an excellent chance that it will not PREP. The default size of integers in FORTRAN 77/V is two words. The default size of integers on the HP3000 is one word. This is where the majority of problems arise.

Another problem is the new logical type. You can not use logical arrays to store non logical data such as IMAGE parameters.

The other road block that is thrown in your path when you are converting is FORTRAN 77/V's implementation of character strings. A character string has two parameters associated with it. The first is the byte address of the string and the second is a one word integer whose value is the length of the string. When you pass a string to a called procedure, FORTRAN 77/V will pass these two parameters. FORTRAN 66/V passes one parameter per string.

II. Integer Data: One Word or Two?

FORTRAN 77/V's default integer size is two words. Repeat the last sentence until it becomes forever burned in your mind. Understanding this will help you in a smooth conversion. HP has provided you with some control on how programs are compiled that will allow you to specify the size of integers. The fool proof ones are the $SHORT and $LONG compiler directives. If your FORTRAN 66/V program uses INTEGER declarations, as opposed to INTEGER*2, then the $SHORT will force the creation of one word integers in FORTRAN 77/V.

Another way to specify one word integers in the above example, is to globally change all occurrences of "INTEGER " to "INTEGER*2". The result is the same as FORTRAN 66/V. Integer constants are another problem. If you specify $SHORT in the code, then all your constants will be one word, except when they are two words. An example (exclamation points are comments in FORTRAN 77/V):

```
$SHORT
       program test
       integer short_integer    !one word integer
       integer*4 long_integer   !two word integer

       short_integer=32000      !the constant is a
                                !short integer

       i=32000                  !the constant is short as
                                !well as the undefined
                                !variable

       long_integer=32000       !even though $SHORT is
                                !declared, the constant takes
                                !on the size of the left hand
                                !side of the equation

       stop
       end
```

The rule of thumb I use for constants is: The constant will take on the characteristics of the left hand side of the equation. This rule also works where there is no apparent left hand side. An example:

```
       program test
       integer*2 parameter,i

       i=32000                          !short constant

       call something(parameter,i+2)  !the second parameter
                                        !is a DOUBLE integer
       stop
       end
```

The left hand side of the second parameter in the procedure call is defined as the parameter in the called procedure. FORTRAN has no knowledge of the called procedure in this example, so the compiler generates a double integer because the default integer size in this code is two words (no $SHORT). The constant in the parameter is a double and it forces the expression to evaluate as a double. If the called procedure expects a short integer then there is a problem. There are two ways to solve the problem. The first way is to force the constant to be a short integer and the second way is to use the $SHORT declarative. Examples are:

```
       call something(parameter,i+2i) !the 'i' following the
                                       !constant forces it to
                                       !be a single integer
                                       !2j will be a double

$SHORT
    .   .
    .   .
      call something(parameter,i+2) !$SHORT forced the
                                     !expression to evaluate
                                     !as a single integer
```

Let me show you an example of an integer overflow:

```
$SHORT
       program test
       integer*4 i
       integer*2 a,b,c

       a=32000
       b=32000
       c=2

       i=(32000i+32000i)*2i

       i=(a+b)*c
```

```
      stop
      end
```

The first expression will evaluate just fine. The second expression, even though identical to the first, will fail with an integer overflow. In the first expression, the constants are defined as single (remember the "i"'s), yet the compiler makes them double because of the left hand side of the expression. The second expression fails because the intermediate values on the stack are single integers. 32000+32000 will overflow a single integer. Inconsistencies in the compiler like these make programming in FORTRAN 77/V a much more exciting challenge than programming in FORTRAN 66/V.

III. Logical Data: Illogical

Logical data types can be described as 'weird'. FORTRAN 77/V only uses the low order bit of the high order word. When the bit is set, the value is true. This is totally incompatible with anything else on the HP3000. If your application passes logical flags around and you must convert only parts of your application to FORTRAN 77/V, then you have a lot of work to do. One way to solve the logical problem is to pass the flags as integers, testing them for zero or minus one. FORTRAN 66/V logicals and SPL logicals are not compatible to FORTRAN 77/V. The default size of the logical data type is two words.

IV. Character Data: Two Parameters for the Price of One

There is a useful FORTRAN 77/V intrinsic function that returns the length of a string. A further enhancement is dynamic strings in subroutines. FORTRAN 77/V will allow you to write routines that manipulate strings of unknown length. An example:

```
      program test
      character string*121

      call string_sub(string)
      stop
      end

      subroutine string_sub(string)

      character string*(*)      !unknown length
      integer string_length

      string_length=len(string) !len is a FORTRAN 77/V
                                !function that returns the
                                !length of a string
      print*,'Length of string is=',string_length

      return
      end
```

There is a drawback to this flexibility. As stated before, two parameters are passed with a string. The first parameter is the byte address of the string and the second parameter is a one word integer by value whose value is the length of the string. This is the only case where FORTRAN 77/V will pass a parameter by value on its own (you can force it to pass parameters by value with the alias directive). Calling a non FORTRAN 77/V routine with strings, or being called by a non FORTRAN 77/V procedure with strings will fail. I will describe HP's work-arounds in a later section.

V. Calling FORTRAN 66/V: Mixing the Old and the New

FORTRAN 66/V's default integer size is one word. FORTRAN 66/V does not have the LOGICAL*4 data type. FORTRAN 66/V expects one parameter per character string. The .true. value of a logical parameter is not compatible between the two fortrans.

If you are careful with integer constants then you will have no problem calling either fortrans. LOGICAL*4 can be passed to FORTRAN 66/V into an INTEGER*4 parameter if you relax the parameter checking.

You can pass character strings to FORTRAN 66/V in two ways. The first way is to use the compiler directive $FTN66_3000 CHARS ON. This directive tells the compiler to pass only the address of the string to any routine it calls. The other way is to define the called routine with an $ALIAS compiler directive. In the ALIAS directive you specify the language of the called routine as FORTRAN 66 then the compiler does the rest.

Calling a FORTRAN 77/V routine from a FORTRAN 66/V routine with a character parameter has the same problem only reversed. You have two ways to solve this problem as well. The first way is for you to provide the extra length parameter to the FORTRAN 77/V routine. An example:

```
C FORTRAN 66/V
      program test
      character string*20
C
C length parameter is by value
C
```

```
     call string77(string,/20/)
     stop
       end

C FORTRAN 77/V
      subroutine string77(string)
      character string*20         !or string*(*)

      string='this is FORTRAN 77/V'
      return
      end
```

The second way is to surround the subroutine statement with the $FTN3000_66 CHARS ON and OFF directives. The above subroutine rewritten in this way is as follows:

```
$FTN3000_66 CHARS ON
       subroutine string77(string)
$FTN3000_66 CHARS OFF
       character string*20  !you can not use the *(*)
                            !construct here

       string='this is FORTRAN 77/V'
       return
       end
```

You may not use the unknown string length definition in the above example. The length of the string is not expected by the subroutine, so the length parameter is not needed in the calling routine. This is the why you can not use the *(*) construct for the above string. This method is the easiest way to integrate 77 code with 66 code. You do not have to recompile the 66 code that calls a routine that you have replaced with FORTRAN 77/V if character strings are passed to the replaced code.

As you can see, calling FORTRAN 77/V and FORTRAN 66/V from each other is not that much of a problem except for logical data types.

VI. Calling System Intrinsics: Is FORTRAN 77/V on Speaking Terms WIth MPE?

MPE intrinsics require parameters that FORTRAN 77/V can not provide; such as value parameters and byte address of logical arrays. HP has provided two mechanisms that will allow you to call system intrinsics and other SPL routines. The nicest one is the SYSTEM INTRINSIC declarative. Ninety-nine percent of your problems in calling system intrinsics are solved with this declarative. The compiler will use the SPLINTR file to set up the call to the intrinsic. The parameters will be passed in the proper manner; by value or reference, by word or byte address. Always declare the system intrinsics that you use in your programs. Failure to do so will crash programs.

The other mechanism is the $ALIAS directive. The ALIAS directive is best used to call SPL routines, although it can be used to call system intrinsics. The directive allows you to define the passing method for each parameter. Based on your definition of the procedure from the ALIAS directive, the compiler will create the code to call the procedure.

You can not use logical arrays to store parameters for system intrinsics in FORTRAN 77/V. Moving a logical array to another logical array will not move all the data from the source array. Only the low order byte of the high order word is moved. An example will explain this better:

```
program test
logical*2 l_array1(10)  !the default size of logical
                        !is two words, so I used '*2'

character string1*20
equivalence (l_array1,string1)

logical*2 l_array2(10)
character string2*20
equivalence (l_array2,string2)

string1='abcdefghijklmnopqrst'
string2='                    '

do i=1,10
   l_array2(i)=larray1(i)
end do

print*,string2
stop
end
```

The output will be 'a c e g i k m o q s'. FORTRAN 77/V will allow you to use integer arrays in place of logical arrays to store your parameters. The system intrinsic declarative will take care of the necessary conversions for you. Do not put parameters in logical arrays, period. The Migration Aid will not place parameters in integer arrays, you must.

The FORTRAN 77/V compiler seems to relax parameter checking at PREP time for system intrinsics. Therefore, you do not have to worry about having a mixture of FORTRAN 66/V and FORTRAN 77/V code that calls intrinsics with different parameter types such as logical arrays in 66 and integer arrays in 77 for the same intrinsic call.

There was one interesting bug that happened to me in a call to FREAD. The length field that I used to receive the value of the FREAD function was defined as a double integer. Immediately after the FREAD call, I tested the condition word. Every time the condition word was CCG, indicating an end-of-file state. I knew this error condition was wrong because the file's record pointer was updated to the next logical record in the file and the pointer was not greater than or equal to the actual end-of-file. The cause of the problem was that the FORTRAN 77/V compiler generated some code that converted the short integer returned by FREAD to the double integer. The code affected the condition word before I could test the result of the FREAD.

Conclusion

Should you convert to FORTRAN 77/V? Yes. Even with the problems associated with a 32 bit language on a 16 bit machine, FORTRAN 77/V is well worth the trouble that it causes. With the new control statements - DO WHILE and IF...ELSE blocks - plus powerful string handling constructs, you can write better programs easily. The FORTRAN 77/V Language allows you to write structured programs without any GO TO statements. GO TO's are difficult to read in programs and they allow for careless programming. The only statement labels that are necessary in FORTRAN 77/V are FORMAT labels. The programmers that have to maintain good FORTRAN 77/V code in the future will appreciate the absence of GO TO's. The string handling helps the language to be a more general purpose language. If only they added BCD. If you are interested in having your code run on the 900 series computers, then you need to convert to 77 because there is no native-mode 66 compiler.

Managing Application Programming With Fourth Generation Resources

N. M. Demos
Performance Software Group
Baltimore, Maryland 21228

In the late fifties IBM realized that its computer effort was fragmented into several different computer models employing several different architectures. At the same time, system software, particularly operating systems, were recognized as an integral part of a computer vendor's offering. To meet this challenge, IBM after much soul searching and internecine warfare, announced the System/360. While electronically different and designed and produced by different engineering groups and plants, each model would have the same instruction set and peripheral interfaces. Therefore programs could be ported at the object code level from one model to another. This would make upgrading much easier for the user and optimize programmer resources, because only one instruction set and architecture would have to be learned.

Software could be designed and written based on the same instruction set. This was a critical element in the strategy and success of the S/300 architecture.

The other radically new feature of the S/360 is that it was designed to be used with an operating system. The operating system, using the new capabilities of the disk drive, would facilitate program to program transition, handle all input and output, and allow multi-programming. IBM initially announced two operating systems for the S/360 - DOS and OS (there was also initially a tape varient of DOS called TOS). OS, designed for the larger and faster S/360 models, was designed to have a functionality way beyond the capability of anything seen previously in a commercially available computer system.

Brooks' "The Mythical Man Month" is an analysis of what happens when a very large system design and programming effort - in this case IBM's OS - is undertaken. While I recommend that everyone read this book, as its lessons are still germaine today and many of its precepts ignored, I can tell you what the message I received from the book was.

Beyond the obvious - man and machine resources - the book has some very important insights on how to bring a large system to completion on time and within budget. As well as showing some of the human frailties that make a project manager's life so difficult, it emphasized that only by superior organization and a thorough understanding of the task could a large project be accomplished. Brooks estimates that a program that interfaces to other programs takes at least three times as much time.(1) He also states that "The man month as a unit for measuring the size of a job is a dangerous and deceptive myth."(2) First, this confuses effort with accomplishment, which are not the same. More important, this type of scheduling assumes that a job "can be partitioned among workers, with no communication among them."(3)

There are two tasks that have to be carefully accomplished for the project to be successful. In the first place, the project must be logically organized and then broken down so that each person has a task that can be understood and accomplished by one person. At the same time all interfaces, vertically and horizontally, must be thoroughly, absolutely and immutably defined. Of course, we all know that that is impossible, but we must get as close as possible. Incidentally, it would seem to be in this area that the Spectrum project may have gone astray.

Once having designed and written specifications, the project must be organized and scheduled. Brooks sees another pitfall here. Ideally, one would like to have a "small sharp team" - a small group that knows the tasks thoroughly, communicates with each other effectively and can perform effectively together. Unfortunately, this group of near geniuses is mostly unavailable and in any case, cannot handle large tasks in a timely manner. Brooks proposes instead project teams of specialists with only one person taking responsibility for all the code, most of which he would write himself.

How have some of the more recently available resources, particularly Fourth Generation languages affected the project director's ability to perform these tasks? First of all, a good fourth GL, because code can be written so much faster and modified more easily, allows realistic prototyping, so that both the user and designer can quickly determine if they are on the right track. This not only verifies the design, but gets the user involved and therefore he is more likely to become committed to the project. Today, with most user interaction occurring via a display, it becomes critical that the users understand what they are expected to do to accomplish the application's objectives. This means that not only must they become comfortable with the terminal, but it must be easy and as obvious as possible to enter data and perform other functions correctly. Prototyping is one way of making this happen. In many cases the last prototype becomes the production version. In other cases, the prototype is a partial solution to the task and is used until the complete system is available. In situations where the prototype or a portion of it will be used in production, the fourth GL and the user implementation must be robust enough to operate in a production environment.

A good fourth GL supports a dictionary. This is a major asset in standardizing data definitions, linkages, and databases so that all programmers access the same data structures in their programs. If carefully managed, not only does this help implement standards and prevent errors caused by programmers not having up-to-date specs, but it facilitates changes. A good dictionary can be used for even more than that. Depending on the dictionary and the skill of the dictionary user, it can store in accessible form definitions on all linkages and program code, thereby allowing members of the project team to query it for as complete a map of the project as they might require.

When a fourth GL is chosen, each one under consideration must be analyzed not only for what it can do itself, but for the environment it operates in. For example, if documentation is an issue, then Infocentre's Documentor supplies a solution not available with the other fourth GL's. If the applications are complex in logic, then the fourth GL must have the capability to accept procedural code and run it on the 3000 without performance degradation. For example, the combination of HP's Transact and Performance Software Group's FASTRAN accomplishes this objective.

The bottom line on project planning is still the same - the better the planning and organization, the better chances for a successful implementation. The availability of new facilities such as 4GL's and dictionaries, if properly utilized, give the project management powerful new tools to help him.

H-P's Precision Architecture - Strengths and Weaknesses

N. M. Demos
Performance Software Group
Baltimore, Maryland 21228

The writer was a systems Engineer with IBM during the introduction of the S/360. The S/360 was a single architecture, implemented in different CPU's, that replaced the existing widely different models of IBM computers. While it established a new standard for all IBM computers, it was radically different in architecture (instruction set) than any of the existing IBM computers (1400 series, 1620, 7040, 7070, 7090, etc.). The parallel to HP's Spectrum project are striking.

The Wall Street Journal labelled Spectrum "the most adventurous gamble the company has ever undertaken." We have all been hearing about Spectrum for a long time. Remember when it replaced "Vision" as the HP 32 bit entrant?

Now that some models of the Spectrum series have been delivered, we can take a look at what the initial results of this five year development have been. The first of the 900 series (which is what HP is calling the Spectrum offerings in the commercial marketplace) was the 930. The Series 930 used Schottky TTL logic to deliver CPU performance of 4.5 million instructions per seconds. It supported 24 megabytes of main memory and dual I/O busses as a standard feature.

The 930 was a stop gap machine. I would almost say that it was a pilot system, which is being replaced by the 950. It incorporates the logical architecture of Spectrum. Its main benefit was a test bed for HP and the very first beta test sites.

The series 950 is the first of the Spectrum series to take advantage of HP's NMOSIII VLSI processor. It will execute 7 million instructions per second. It will have up to 64 megabytes of main memory. It is the first HP product to marry the architecture with new chip technology. In order to take advantage of the improved performance, users will have to recompile into native mode and convert to Turbo-Image if they have not already done so. This means that their source programs must be either in COBOLII, Pascal or Fortran 77. Those are the only native mode compilers that are available at first release of the 900 series.

HP has taken the concept of reduced Instruction Set Computing, adopted it and added other new developments in hardware and software design. They call the result HP Precision Architecture. The basic features of Precision Architecture are:

1. Reduced Instruction Set
2. Fixed-length and fixed format instructions
3. Load/Store design
4. Hardwired instructions

5. Single Cycle operation
6. Optimizing Compilers

Reduced Instruction Set Computers (RISC) are the current darlings of the computer industry. In a netshell, this engineering philosophy states that a well chosen simple set of instructions that can be hard wired is better than a much richer more complex set, usually implemented in microcode to keep the cost reasonable. First of all the CPU is much easier to engineer and therefore can be designed economically to run at much higher speeds. Secondly, well designed "optimizing" compilers can be more effectively implemented for a RISC machine than the previous CISC (Complex Instructions Set Computers) because the compiler writers have fewer instructions from which to choose. The rationale for RISC is that changes in circuit technology, the speed and cost of accessing memory and better high level language compilers, have made the CISC machine technologically obsolete. Memory access speeds have increased faster than CPU speeds. RISC, with fixed length, fixed format instructions and single cycle operation is inherently very fast for most operations that have to be performed on today's computers. However, what is "Reduced" is not defined. There are about 127 instructions in the 900. I can remember computers that had only 1/5 that many instructions. The HP implementation of RISC is not that "reduced."

Other elements of the 900 Series are also a radical departure from current 3000's. A virtual memory addressing scheme is used whereby disc memory is regarded as an extension of main memory as required. This scheme fits very well into the caching scheme used by the 900 Series and is a very effective method to access both data and instructions. However, it is a radical departure from the current methods, although it eliminates the dual buffering inherent in a cached 3000 system today. Obviously, the entire I/O system must be re-written to support this I/O method.

The software is the critical element in the 900 Series. Not only is a huge amount of software required for the operating system, which goes under the name MPE/XL, but the new optimizing compilers require an even higher level of technological competence than previous compilers. At the same time HP is attempting to integrate their IMAGE database system with their new SQL relational database. On top of that they are committed to run almost all existing code in compatability mode. All this is a huge undertaking. Just doing the new XL version of MPE is a larger undertaking than HP has accomplished up to now. It is the software area that causes many people to have doubts about HP's ability to deliver the 900 Series, with reliable software that gives the promised performance. There is no question that HP will be able at some point to deliver a quality 900 Series product with a reliable, complete set of program products, operating at the promised performance levels. However, today they are a long way from this goal. Getting there is no easy matter.

As an interim step to gaining improved throughput on the 900 Series without having to recompile, HP has something called the Object Code

Translator. This will gain an estimated 10% to 30% speed improvement, and is meant to assist users in cases where they cannot recompile. If it is reliable, it will be of use to users with SPL code.

What does the market think of this radical new product line from HP? There have been several comments that indicate that while the 900 Series is impressive, it has been a long time coming. "HP is currently comparing Spectrum to (DEC's) 8600. By the time the systems come out, they will be competing with an entirely new machine," said Michael Murphy, co-editor of the "California Technology Stock Letter."

HP success with this new series seems to depend on two factors.

1. How long will it take HP to stabilize the new product at a high level of performance and with a complete software offering. If this can be done relatively quickly and as promised, then HP has laid the groundwork for an architecture that will benefit them for years to come. If they are slow to implement their excellent strategy, their credibility will be damaged.

2. What will the competition do in the next year or two to counter and perhaps improve on these new offerings? HP has no lock on any of this new technology. It is known that every other major vendor is looking seriously at RISC and other new technologies to improve price performance. Some have already produced RISC machines.

Where does this leave the HP 3000 user who is looking at Spectrum as an upgrade for his system? The series 70 will be a better choice for many at least in the short term. There are two classes of 3000 users who have an immediate interest in the 900 series.

1. Those users who need CPU power beyond that available on the Model 70 and require that it be on one computer (because of the size of a database, perhaps).

2. Those in large corporations with many 3000's who can afford to bring a 900 series in-house as prototype for future systems.

Those who are not in one or both of the above categories should probably upgrade to or acquire additional model 70's.

Having made the decision to go to a 900 series, the user must develop a detailed migration strategy. He must balance the conversion effort of going to native mode with the benefits. Only native mode processing will give him cost/performance benefits that justify the acquisition of a 900 series. Even programs to be run in compatibility mode need to be tested at least until there is some confidence that compatibility mode is bug free.

By adopting a radically new architecture, HP has taken a giant step toward rationalizing their computer line for future growth. It is an excellent, bold, decisive strategy on HP's part to increase their market share in the long run. The immediate problem for the current HP user who needs more processing power is how to get from here to there. It will be a while before the 900 series stablizes into the kind of super reliable system we have come to expect from HP.

We already see evidence of HP's move to use HPPA to aggressively increase its market share. In the last few years most of HP's computer revenue has come from its installed base. With HPPA, HP now has the hardware (or can soon have) to compete at all levels of computer power. HP has a competitive advantage. For example, it is estimated that HP's HPPA computer can be produced for one-quarter the cost of a DEC CPU of equivalent power.

Hardware is only part of the picture. Software and connectibility ("Networking") are the other parts. Both DEC and IBM have announced major new networking products recently. HP has also, but is having to run to catch up. It is caught in the position of having to do two versions - one for the MPE/V machines and one for HPPA.

It is the software area that causes the HP friendly computer user community the greatest concern. MPE/XL is still not a proven product, although we all hope it soon will be. Realistically, I expect that HP will take some real heat on the software side before things improve to the point where MPE/XL's reliability and performance will match MPE/V. As far as new customers are concerned, HP has taken a step in this direction by targeting an HP/UX "commercial" market. The HP implementation of UNIX has seemed to have been easier to implement than MPE/XL. The 800 series (HPPA-UNIX) computers have been well received in the marketplace. It is estimated that the UNIX market will grow at twice the rate of the general computer market this year. Whether UNIX will become the true standard operating system for the future remains to be seen.

The other problem HP has is the lack of a large repertory of application programs for HP computers. In attempting to increase market share, i.e. attract new customers, software availability is very important. This may slow down HP's ability to attract new customers who are not interested in UNIX.

The next few months will be very interesting for all of us involved with HP.

What Will Programming Be Like in 1998?

N. M. Demos
Performance Software Group
Baltimore, Maryland 21228

Programming has come a long way in the last ten years. By trying to look at the next ten years, programmers and managers can assess the skills and tools they will need in order to be productive in the next decade. 4GL's as currently implemented on the HP3000 today are only the beginning. The ideal programming methodology of the future will include a very powerful application generator that will. (1) Be able to draw on all the previously written application code where appropriate. (2) Be able to provide the programmer with easy to use design templates that he can use to easily generate his program, including screens. (3) Be able to build and modify databases. (4) Be able to automatically create utilities to maintain the databases. (5) Be able to check every programmer action for internal consistency as he enters it. (6) Be able to help the programmer generate test data. (7) Be able to provide all levels of documentation automatically, tailored to the installation's requirement and standards.

For the first time since the advent of assemblers, there are things happening to radically alter the way programming is accomplished. Although these tools are not complete yet, there is enough available so that we can guess how these tools will operate. The programmers and, particularly, the programming managers and MIS directors who fail to learn about these concepts and tools will be ill prepared to perform their tasks in the not too distant future.

One dictionary definition of programming is, "To provide a computer with a set of instructions for solving a problem". That is a very broad definition and would cover such trivial tasks as setting up Job Control statements, etc. We think of programs as not only solving problems, but also as accomplishing a non-trivial task and being easily repeatable. Up to this time most programs have been coded by specialists in programming, whether in-house or purchased. When one considers the computer tasks that need to be accomplished, it becomes clear that quantum leap in program productivity is required. This will require a combination of the following:

1. Much more extensive use of purchased software. This means that the users may have to be more flexible in their requirements and that the software will have to be better and more adaptable to the individual user environment.

2. More user "programming". New tools and an increasing computer skill level will make it feasible for the user to do the simpler programming tasks, such as data extracting and formatting for reports.

3. More sophisticated programmer tools. New tools, of the type that are just beginning to appear on the market, will make the

initial programming task and maintenance programming much more productive.

It is to this last point (and partially to the second) that this paper is addressed.

There will be new methods of communicating with the computer, such as voice recognition, that will make the programmer's job easier for his own interaction with the computer, but may make programming more difficult, because the user will also want to use this technology. We do not see the major productivity improvement being in this area. Rather, we think the major new tools will employ similar techniques that have been used to the benefit of the user. Using data bases to organize and store data relevant to the programming task, i.e. data dictionaries, is just coming into wide use. The dictionaries are getting more sophisticated. They can or will soon be able to store in an organized fashion not only information describing the data structure and location of data fields, but also modules of code. They will be active versus passive dictionaries. That is, changes in any definition in the dictionary will trigger a change to all related (dependent) definitions where this relationship is known.

Application generators closely integrated with dictionaries will give the programmer the ability to conveniently extract all relevant previously done programming work and integrate it into his current program. At the same time, it will be easy for him to enter his new program modules into the dictionary for reuse by him or use by others.

The application generator, to be most effective, will be front ended by a very well designed and integrated prompting and editing system. It will include the capability to interactively run his code and have an extensive debugging system that he can move in and out of easily. An extensive help system will save time searching through reference material. Syntax checking will occur on the fly, so that corrections can be made immediately. Internal consistency checking will prevent the entry of unused or redundant code. Extensive prompting will optionally be employed to assist the programmer in insuring that all necessary parts of the program are completed.

This type of facility will not only apply to the program itself, but also to its production environment. The application generator will already have, as part of the information necessary for programming, most of the information needed for execution of the program. This will not only apply to the program itself, but to the entire system of which the program is a part.

If we look at the way a programming task is accomplished in the future, we can envision someone sitting at a display with a keyboard and voice facility. It will probably have a mouse associated with it. The user of this system might very well have on the screen what we refer today has icons. He will manipulate data, code and tasks interchangeably, to fashion the solution he needs to complete.

In this new way of doing things, not only will there be only one occurence of each piece of data, but data derived from other data will be defined as such, so that a data definition may consist of algorithms operating on the basic data. Information will always be based on the most recently input basic data.

This methodology is technically feasible today, but there does not exist a comprehensive system that can be used now to achieve this kind of environment. Most of the things discussed above involve applying the present technology not only to the organization and retrieval of the data itself, but also to the algorithms used to derive it.

There are already several systems in place that are significant steps in this direction. A prime example is the system used on Apple's Macintosh.

The kind of systems we are describing here have another advantage that may not be obvious. They will impose on the programmer a discipline that will make his programs much easier to maintain, particularly by another party.

HP's recently announced Virtuoso is an application generator that is an example of a tool that will give the programmer the capability to integrate into a new program modules of source code previously entered into a dictionary. Although HP is supplying Virtuoso with a "sample" library of COBOL routines, the programmer can build his library with source modules in any programming language. Virtuoso is meant to be a tool to build a large application system with standardized modules integrated in a standard way. Thus it is an example showing one of the capabilities mentioned above for the new programming methodology.

The New Wave environment is another facility that shows us how programming will be accomplished in the future. "The HP New Wave environment provides the tools that support the user's everyday function and it is also the user's view into the entire office systems network." New Wave looks very much like Apple's MacApp for its Macintosh computer. The concepts of both MacApp and New Wave envision the following concepts:

1. Object oriented programming. An object is a piece of data, a collection of interrelated data or a task operating on data.

2. A common user interface. All programs will have the same look and "feel" to the user.

3. A common structure. All programs will be designed to the same standards and employ similar methodology. The techniques in this area will be similar to the application generators mentioned above.

4. Data and code will be stored only once and be employed as appropriate throughout the system. For instance, if the user changes a spread sheet entry, not only will the affected numbers on the spread sheet change, but the correct effect will be automatically made to any other place in the system where data is used or derived from the changed entry. An object may be not just a data item, but a combination of data items and/or algorithms that operates on the data.

The goal of all this is to give the user an integrated system. HP uses the word "seamless" to describe this level of integration. That is the user will not be aware of running a particular program - he will be accomplishing the task in the most effortless, accurate way possible. At the same time, the programmer will be able to concentrate only on completely unique code and data elements for new applications or application extensions. If these new techniques work correctly, he will be able to be confident that any change to a routine will automatically be propagated throughout the system. Of course, any bug he introduces will also; powerful tools can sometimes result in powerful errors.

STAFF TRAINING, WHY, HOW, & WHEN

Charles H.R. Volz,
VOLZ Associates, Inc.,
34 Undine Avenue,
Winthrop, MA 02152

(c) Charles H.R. Volz 1988

ABSTRACT

A knowledgeable staff is very important to any operation. However, few organizations have a systematic approach to training their staff. This paper and presentation will talk about why continuing education is important and its benefits. We will talk about what is available, how to choose from among the various options, and when to train. We will also discuss how to attract and keep staff with training.

1. IS IT NEEDED & WHY?

I have a quote from Susan Boyd of PC Concepts. She said, "Training is the investment you make in people so hardware and software pay off".

Staff training should be a continuing program for every company. Without a training program employees become stale, complacent, and frustrated. An employee's lack of knowledge can hurt an organization by costing money and time. Lack of training can waste an employee's time. It can waste the investment made in equipment, systems, and software.

Most people want to learn more about their job. They want to tackle new challenges. They want to do things correctly and efficiently. They want to be proud of their work. Training makes it possible for people to do these things.

For a company, employees who know their jobs thoroughly will be more effective participants in reaching the goals of the organization. It can cause morale to be and remain high. Employees are more willing to take on new assignments knowing that training will be provided. There is more efficiency. Goals are met or exceeded.

2. WHEN TO TRAIN

When should a company train? For new employees when they start. No matter what level of expertise the new person may have all new employees should be given an extensive orientation. This orientation should cover department and company standards and procedures. The employee will then be comfortable with following them.

Employees coming into a situation where the equipment or system is new should be trained. The training should begin as soon as possible after the start date. On the job training should be discouraged.

All companies and departments are starting new projects or bringing in equipment. A lot of these things need the employees to learn new skills. Training is very important in this area for a smooth integration of equipment and completion of the project.

Another area where training is needed is when an employee's performance is poor. Training may give the employee the boost to get the job done efficiently and effectively.

Some employees ask for training. They have perceived a lacking in their own knowledge. Or they have glimpsed at a new technology that might benefit the organization. These requests should not be dismissed out of hand. Rather management should evaluate each request and act on its merits. The employee just might be right.

A final area where training is essential is when an employee is given a promotion or a new assignment outside of their current expertise. Training will help give the company the payback it is looking for with this change.
The effectiveness of the employee will be improved if the employee understands what is being asked of him/her.

On the job training should be discouraged. This is the costliest way to train. On the job training is a very slow way to learn. The employee is frustrated and or discouraged. The employee feels unproductive and is unproductive.

If, per chance, on the job training does work, the employee learns enough to function adequately, most plateau. It takes a lot of initiative on a persons part to self train themselves. When this happens usually only what is absolutely necessary for the job is learned.

Are we saying that no self-training or on the job training should be done? No. There are times when this method is very appropriate. It depends on the individual employee. If allowed it must be monitored.

3. WHAT IS AVAILABLE

There are various method for getting employees trained. In our HP 3000 environment the biggest place to look for training is from Hewlett-Packard. Another place is the software developer whose package a firm is using.

For other public classes there are two more organizations to look into. The first is independent firms local and national. If on a HP related subject, many times these firms are more cost effective than HP or the large software suppliers. Most have worked with the subject matter for many years. They may even be more "qualified" to teach than the supplier. Non HP related seminar companies would be more generalized. But most of the instructors used are considered experts in their field.

The other place to look is colleges. For generalized training they can be excellent. Elapsed time is longer with a college, however.

Many of the same above will provide in-house training. The benefit of this type of training is that it can be tailored to an organization using their own data. Easy follow-up and monitoring of employee progress are two other benefits of in-house training. For large numbers of employees it is also very cost effective.

Also for companies with on going internal training hiring a staff trainer is very effective. A staff person gets to know the staff and the company very well. Training can then be tailored to both the company and the employee.

One caveat though; treat an in-house training session as sacrosanct. Meaning do not interrupt unless sickness, death, or public safety forces you to. If you do the investment in training will be lost. And all lose.

In electronic training there are video, audio, computer based training, and interactive video a combination of video and computer based training.. Although generalized, these are self-paced; the employee can move forward when ready. Also time away from work is limited and the courses can be reused saving money.

However, with electronic training, an instructor is usually not available. As a result questions can not be asked. The hand holding support is not there to help smooth rough areas.

A final area for training is conferences. Much information can be learned from these meetings. Not just from the lecture sessions but also from the informal discussions during breaks. Also local/regional professional groups can offer a learning experience. (User groups, ASM, DPMA, etc.)

4. HOW TO CHOOSE & WHO TO CHOOSE

Training is only a good as the people providing it. The criteria listed below comes from research Lotus Development Corp. did prior to setting up their Authorized Training Center Program.

4.1. Price

It should go without saying; cost versus value needs to be evaluated to get the best return. It makes no sense to spend a lot of money on training if the people doing the training are not knowledgeable in the subject or are not comfortable making presentations. This also applies to the major software suppliers and HP. An inexperienced person should not be teaching others.

4.2. Commitment

Will the firm doing the training be around for the long run? (This is also important for software houses but that is another issue.) It is very important the firm doing the training not be a "fly-by-night" firm. The company needs to be committed for the long run. The company should have a proven track record for training.

For the in-house programs, again the commitment to stay in the marketplace is important. If the seminar is going to be tailored to the client, the client should expect the development of the course to take time. Also the client should insist on confidentiality.

4.3. Strength of Relationship

You should not be afraid to call on the firm giving a seminar to discuss content, ideas, methods, or subject. A good relationship will ensure better programs, more success for the employee/firm attending a seminar, and, of course, increased productivity.

If the relationship between the trainer and trainee is not good, the value for the attendees goes down. This is not to say that it is necessary to know the firm doing seminars or the instructor completely. Just that the opportunity to discuss ideas, etc. is there.

For in-house seminars, the willingness to customize a seminar to the clients environment may well show a willingness to develop a strong relationship. Customization may not be applicable to every course offered. But if it is, the use of a clients own data, environment, and/or vernacular will help them get the most out of the course.

4.4. Training Off- or On-site

For those attending a seminar it is usually better to have the training off site. This is true even for the "in-house" seminars. The reason is that the students are away from interruptions. They can concentrate on the subject being presented to them. It matters not whether the firm giving the seminar has there own facilities or use public facilities. For that matter space on the client site is acceptable if the client management refrains from interrupting. The only immediate reason for interrupting is for death, sickness, or safety. Any other interruptions should be thought out carefully before being done.

4.5. Timing

Naturally, the best time to train is before the need is critical. The scheduling of a training session should be made with this thought in mind. It makes no sense to train someone in tax preparation after the tax season. Unless it is in preparation of the next season.

4.6. Resources

Resources is a broad category. Here we mean class size and equipment for student use. Resources are also determined by how "hands-on" the class is to be.

The number of students per instructor depends on the type of program being taught. Very "hands-on" should be less than ten students. A very interactive session with questions should be kept below thirty. A lecture has no limit other than what the facility can handle.

If equipment is being used in the learning process, i.e. a PC or terminal, it is best to have a one-on-one situation, one student to one PC/terminal. That way they can learn from there own mistakes. Certainly no more than two students per PC/terminal should be allowed.

4.7. Product and Industry Expertise

The expertise of the instructor is very important. Is the instructor well known in the field? This does not necessarily ask if the instructor has published a lot of papers; but what is the instructor's reputation.

Length of service in an area is also important. But while evaluating the number of years experience, keep in mind how new the technology is. For example, MPE-XL classes will begin in another year. No one has more than two years experience with it. In this situation it is better to look at the instructors overall experience and if they have taught before.

Teaching experience is very important. One maybe an "expert" in a particular field but be unable to teach.

4.8. Levels of Instruction

While it is not advisable to have different level of experience in a class, sometimes it can not be helped. It is, therefore, necessary to insure the instructor can handle experience levels. It is also necessary to look at the level of experience the seminar is being aimed at. Prerequisites may preclude various levels from attending a class.

If multi-level experienced students are allowed and the instructor can handle them, it is necessary to have appropriate material for all the levels.

4.9. Materials

For any type of educational session, presentation materials should be made available. Vendor specific sessions should provide the presentation, tutorial, workbook and a reference guide. Additional support material such as glossaries, articles, and technical tips are helpful.

4.10. Follow Up

Follow ups run in many formats. The very least should be a willingness to send answers to questions that remain unanswered at the end of a session. The offer of a subscription to a newsletter or information reporting service is helpful. Also helpful is a follow up session to advance students along.

Another area in follow up is a willingness to ask for session evaluations and critiques. This is a very important indicator of a trainers commitment to a long term service.

5. HOW TO PICK A COURSE

5.1. Audience

Picking a course is as important as picking who to teach the course. The first criteria to look at is what audience the course is aimed for. This does need careful attention because the nomenclature used can be vague. Just as it makes no sense to send a person to an advanced system manager class if they have never been to a system manager class. It is equally wrong to send a person to a class titled "New MPE Issues" if it is really an intro to MPE. Above all, avoid the general good for all types of attendees. No one benefits from these classes.

5.2. Prerequisites

Audience prerequisites should be measurable. Actual skill levels should be stated clearly. Statements such as 3 months experience or if familiar with such and such are not clear enough. Measurable means can the attendee can perform at this level or do a particular task after taking the class.

5.3. Performance

The sessions performance oriented objectives should be clearly described in measurable new skills. The objective should list who learns, what will be learned. It should be explained under what conditions the student can accomplish the goals; what level of performance the student can achieve. Particularly helpful is a statement on how to measure for the goals.

Measurability will be shown if the class brochure states clearly these issues in action words rather than passive words. For example, words such as do or write are action words. A passive word is understand.

5.4. Support material

When choosing a course look at the outline of content. One should always be given. It may be concise. But the outline should be descriptive of the course content. If case histories are appropriate their content should also be described.

5.5. Practice and feedback

Practice sessions can be a very important aspect of training. Some classes by their very nature and brevity preclude them. But it may be something that fits your situation. If practice sessions are included check on whether testing is a self test or a proctor given test. Also check on whether the results of testing is immediate. Immediate test results should mean quick correction of errors.

Question answering is also important. Most questions should be answered immediately. If the question was ahead of an issue then reference should be made again about the question. Questions should be encouraged even in self-paced classes.

5.6. Review

We mentioned review earlier in terms of evaluation as a whole. Here we what is asked is if the material can be reviewed easily by the student after taking the class. If a case study is provided can it be used for later review.

5.7. Self-paced training

For self-paced training you need to know if the student can focus in on the material quickly after being away from it. Also, the course should be designed in a modular manner to allow for frequent and easy review of material before going on.

5.8. Other

For on-site classes customization should be readily available.

6. HOW TO GET TO A TRAINING SESSION

Justification is the key to getting to a training session. One must prove the benefit of going versus not going. If the training session meets the criteria stated above, there should be no problem in justifying.

For some other tips on how to get there. If a potential student proves to him/herself that the session is valuable they ought to just go. Invest in his/herself; take the time off and attend. The student will get the benefit and it makes no sense to deny oneself because an employer won't spend the money. (An employer might actually have more respect for the person for going on his/her own.)

Another way which is good when getting in new equipment or a new system is to build it into the purchase order. If possible, note the dates of the session on the PO so that all is approved at the same time. Having the training in the original PO avoids the chance of refusal.

7. WHEN NOT TO TRAIN

Giving classes out as an award for job performance or any other reason actually negates the reason for training. Everyone wants to do a good job. They want to grow. Training should be looked at as an investment with benefits and long term saving being reaped by the company.

To give someone training in an area in which they are already skilled makes no sense. And yet it is done by well meaning people. There is no benefit to the company or the employee. It would be better to just give the trip without the class and call it an incentive award.

It could be argued that sending a person to conferences and local/regional user group or other professional group meetings should be an award. And sure there are social activities at these meetings and conferences. But there is also valuable information available which when brought back benefits the company. At the very least a rotating schedule of employee attendance should be arranged so that all can grow. This also ensures that the company gathers all the information from all points of view.

8. CLOSING

Training is a luxury no organization can afford to be without. Training stimulates employee growth and company growth. Training causes a synergism by the sharing of ideas and knowledge. An exponential factor comes into play when there is an on going training program benefiting all as a group much more greatly than by themselves.

The education process must never stop. It has a much shorter life-cycle in today's fast changing technical world.

COMPUTER AIDED PUBLISHING

Charles H.R. Volz,
VOLZ Associates, Inc.,
34 Undine Avenue,
Winthrop, MA 02152-1431

Copywrite 1988 Charles H.R. Volz

Computer Aided Publishing has been around for a number of years starting with computer aided typesetting and word processing. Desktop publishing or DTP is what we hear a lot about today. But DTP is only a part of Computer Aided Publishing. CAP encompasses more than just desktop equipment but also large systems such as IBM, DEC, and HP.

Corporate Electronic Publishing is also another term used; as is Electronic Publishing. The all CAP, CEP, and EP say the same thing.

WHAT IS IT

Electronic Publishing is the use of computer tools to produce and distribute publications.

Publish is defined in the American Heritage Dictionary as "1. To prepare and issue (printed material) for public distribution or sale. 2. To announce to the public." Publication is defined as "1. The act or process of publishing. 2. Published material. The word publish come from the Latin verb - publicare - to make public.

The process of publishing includes the design composition, page make-up, printing, and distribution of a readable item.

BASIC COMPONENTS OF CAP

There are three (3) basic components of CAP, the input system, the composition system, and the output system.

Input systems are for creating and editing text and graphics. The "inputted" material is then processed by the composition system for page layout. Input systems include:

Text -	Editing with word processing software; dedicated word processing systems; Optical Character Recognition systems (OCR); turnkey publishing systems.
Graphics-	illustrating, drawing, CAD, Presentation software; OCR equipment that allows for electronically produced and stored graphics.
Database-	data needed for integration into a document.

Composition systems are the base of any CAP system. Composition systems take the input as described above and compose or layout the page or pages as directed by the user. The composed page is sent to an output device or stored.

The capabilities for composition systems are:

- the storing and manipulating of data, text, and graphics;
- output prepared for laserset or typeset copy;
- intermixing type styles and type sizes as selected;
- performing pagination functions automatically for organizing a document into a page or pages or other fixed format ready for output.

Output systems are usually hard copy devices such as laser printers and typesetters. Twenty-four (24) pin or better dot matrix printers can be used but the type quality is usually lacking.

Output systems can also be electronic in the form of CD-ROMs, vidoetext, and on-line retrieval systems.

INTEGRATION

Most times the three components input, composition, and output are from varying manufacturers. This means that of all Office Automation systems, CAP systems require integration capabilities. CAP systems need to be able to take a document with any format codes intact and send it to the composition system. The composer (composition system) should use the format codes sent plus add any new codes needed to compose/layout the document. The composer should then send the document to an output device (ex. printer) in a format usable by the output device. The end result should have the document appearing the way the user wanted it.

In order for the composer to do this it is desirable for it have software filters. Software filters allow the composer to retain the content and design of a text and/or graphic file created by another system.

The composer should also be compatible with existing standards and with existing leading products. It should have the ability to link to PC based solutions.

Lastly, the composer should fit within the corporate culture using it, the network environment, and be flexible enough to incorporate new technologies or other future changes.

CAP CATEGORIES

CAP systems fall into two categories, the dedicated single or multi-user systems and desktop publishing.

Within dedicated CAP systems are two sub-categories. The first is the centralized CAP system with many terminals attached to a single processor. This is much like a mainframe environment. The second is the decentralized, distributed systems on a network or a stand alone system. Many of these CAP systems are UNIX based.

APPLICATIONS

Before I go into desktop publishing I want to cover some of the applications that can use CAP.

Some people try to divide CAP into two camps segregating DTP production from dedicated production. And in some cases the separation is good. Yet in the simplest of terms both categories satisfy the two basic application types :

Corporate Publishing &
Commercial Publishing.

The differences between the two are based more on audience than quality.

Commercial publishing is the creation of very high quality documents for profit. The examples are books and magazines. The audience most times is the general public. Newspapers are also of commercial publishing but the overall quality is lower.

The corporate publishing audience is usually the corporation or its clients. Quality of output can be very high but for most items is not high. Corporate publishing has three sub-categories-

- General office and Marketing
- Technical
- Financial.

Taking the less lengthily sub-category first, financial publishing is a unique area of publishing. It is the only one of all the sub-categories that need the ability to make last minute changes just before printing. Usually, there are many approval steps and cycles necessary both within a company and outside do to various financial reporting regulations. Also, financial publishing almost always includes tables. Types of financial publications include -

- Prospectuses
- Quarterly reports
- Annual reports
- Mutual funds
- New stock offerings.

Technical publishing is the category of publishing where a CAP system cost justification is the easiest. Technical publishing requires the ability to continually update, add, or delete information from a document. Many times charts, graphs and computer aided designs are part of a document. There is a need for a single original with many people having access to it.

Technical documentation / publication departments tend to use centralized stand-alone CAP systems. Types of technical publishing are :

- User manuals
- Reference manuals
- Service manuals.

The general office and marketing sub-category is the fastest growing area in publishing systems. It has been estimated that a typical Fortune 1000 company spends between 6% to 10% of gross revenues on publishing. Savings realized with CAP are estimated to be as much as 50%.

The kinds of documents that fall under general office and marketing are -

- Internal reports,
- Requests for Proposals / Quotes,
- Business plans,
- Marketing plans,
- Sales literature,
- Sales proposals,
- Financial reports,
- Newsletters,
- Bulletins,
- Personnel documents,
- Company policy manuals,
- Forms.

DESKTOP PUBLISHING

Most of us think of desktop publishing first when we think of CAP. For unless you are in a commercial environment or have a large technical documentation department, multi-user or solely dedicated CAP systems are not looked at.

DTP systems will handle the general office and marketing requirements as stand-alone, dedicated, networked, or intelligent workstations. And, most importantly, the output can now go to a variety of typesetting equipment. (IBM has come out with a plate making device which will connect with a PC.)

The term "desktop publishing" was coined about three years ago by Paul Brainerd, president of Aldus Corporation which developed and sells PageMaker. He developed this term to describe the use of off-the-self PC components. Mr. Brainerd and others of Aldus came from ATEX which has been in the business of CAP systems for newspapers.

The typical set up for DTP is a PC, an Apple Macintosh or MS/DOS 286 chip based compatible. The input software consists of a word processing package, a graphics or drawing package, and a spreadsheet package. The composition software is then added. Finally, there is the output device usually a laser printer.

The PC's memory should be 640kb to process the large amount of data used. Also on the IBM compatibles the programs are memory resident. This means staying in memory even if in a background mode. For disk drive a 20Mb hard drive is the minimum.

LASER PRINTERS

Laser printers give laserset output. Laserset is near typeset quality. The technology involved in producing laserset output is similar to photocopy technology. Toner or ink is drawn to a magnetized drum, then transferred to paper as the drum rolls over it, and lastly, melted onto the paper for durability. The technology allows for the printed page to appear very clear.

Software languages called page description languages (PDL) describe to the output device, laser printer, how a page should look. The description is in terms the output device can understand and produce. The PDL's tell the device how wide a line is, where to place the information being printed, what a character or letter looks like in size and style. PDL's require graphics capable printers like the laser printer. They very simply "plot" each character and page. All this is very much transparent to the user.

PDL's uses typefaces or fonts to produce characters. The fonts indicate the size and style of a character. It also tells it the character is to be in a bold face, italic, or neither (normal). The fonts are designed by type designers. Some are in the public domain not requiring licensing. Other typefaces/fonts are copywritten requiring permission or licensing to adapt them for the PDL's.

The two best known PDL's are Adobe's PostScript, considered a near defacto standard with interfaces to laser printers and type setting machines. The other well known PDL is Hewlett-Packard's Printer Command Language (HP-PCL) which by virtue of the large sales of LaserJet's and emulators is the most used. (Some people would argue that HP-PCL is not a true PDL.) Other PDL's are Imagen's IMPRESS and DDL and Xerox's Interpress. There are differences between them which is the subject of another paper. Suffice to say, for the most part, they should be transparent to the user.

COMPOSITION SYSTEMS/SOFTWARE

At the beginning of the paper, we mentioned composition software/systems in broad terms. In more definitive terms page composition software allows you to :

- Layout the page the way you want to see it,
- take text and change the type style, type size,
- aid in cutting and pasting a whole page,
- set up master page formats to be used repeatedly within a document,
- set up templates to be used with many documents,
- run text around pictures and illustrations easily,
- text flowing, the automatic flowing of text to the next page or column,
- kerning, moving letters closer together so a word looks better,
- Automatic leading, the spacing between lines of text,
- tracking, the placing of space around a grouping of characters,
- paragraph spacing, the adding of space before and after a paragraph,
- and more.

All to make the finished product look better and be more readable.

There are two philosophies to composition software. One is the interactive designing, composing, laying out of a page. The infamous WYSIWYG or What You See Is What You Get. Within video display limitations and, for the most part, the lack of color printed output, this is true. The other philosophy is embedding description codes within a document file that has the text and graphs merged. The document is then processed in a batch producing the output. The differentiation is getting hazier as the batch systems appear more like WYSIWYG.

Examples - PageMaker is WYSIWYG
- SGML (Standardize General Markup Language) is batch.

GRAPHICS

We also touched on graphics as being a component of the input system. Graphics actually comes in many flavors for DTP. The base is presentation graphics, the pie charts, bar graphs, linear graph, overheads, etc. You can now do much more.

There is now software called drawing packages that let you "freehand" draw a chart or picture much like a pencil drawing. To take this a step further, there is also software called paint packages. This software allows one to paint a picture with full color using a plotter. There is also Illustrator from Adobe which give the precision an illustrator needs to produce clear crisp illustrations using PostScript and a laser printer.

TYPESETTING / LASERSETTING GUIDELINES

Awhile back, we talked a bit on fonts. In this section we will give some guidelines on how to assure professional looking, neat, and readable documents.

Typesetting is an old art. It is the art of placing each letter on a page so that it is attractive and readable. The old manual method is still practiced. The artistic value of hand set documents is very high.

In this old method each character is carved on a piece of metal, usually lead, in reverse and raised much like a typewriter key. These metal pieces are then placed individually in a frame. Spacing between lines is with a blank piece of lead leading to the term "leading". Once the frame is filled it is inked and "pressed" onto paper.

We have gone beyond the hand set method now with photo-typesetting. In photo-typesetting a film negative copy of a document is produced. Then a metal plate is made by acid burning it to replicate the negative. The plate is the inked and copy "pressed" on to paper.

With typesetting / lasersetting a document can be produced with much smaller type than the standard letter quality printer or typewriter produces. And the document will still be attractive, clear and readable. There are some guideline that need to be followed though if a document is to be truly readable, professional, and neat.

The number of type styles used should usually be kept at two. Certainly no more than three. If any more type styles are used readability drops and a document appears cluttered.

Avoid using incompatible type styles. There are many typefaces available and many headline typefaces. But that does not mean that when mixed, styles will blend with each other.

Once you have selected a size and style for the text body stay with it. The same is true of headings, sub-heading, and footnotes. Body text size is usually between 8 - 12 pts. (14-10 pitch). Headings and sub-headings are larger; footnotes are smaller.

Be careful with the over use of emphasis - the bold, italic, and underscore. To much use of emphasis means nothing gets emphasized.

For page layout -

- Balance the page by keeping proportions and symmetry in mind when placing text and graphs.
- Leave reasonable margins of between a .5 inch to an 1 inch on the right and left sides, .75 inch on top, and 1.25 inch on the bottom.
- Keep lines at a reasonable length. A range of 40 to 60 characters is what the experts say is readable.
- Be consistent with the number of columns used in a document. Also, place the columns the same through out a document if the column widths differ.
- Keep headlines on the body of the text being headlined.
- When using graphs, keep them as simple as possible and place the appropriate graph near the text describing it.

BENEFITS

There are many benefits to CAP whether or not the actual printing is done in-house. Below are some of them. But remember when a CAP system is selected, it must fit in to your company's environment, datacomm, work flow, and philosophy. Also remember that not everyone is capable of utilizing CAP. Fundamental design basics must be either learned or come naturally to the person operating the CAP system.

Back to benefits. Some of the benefits of a CAP system can be easily measured. In this world of bottom line thinking, fortunately cost reduction is the easiest.

With CAP the costs of typesetting services, graphics services, and layout services can be reduced. Bringing these duties in-house can virtually eliminate the need for an outside service.

Paper supply costs can also be reduced. Typesetting / lasersetting can actual reduce paper consumption because a smaller type can be used and still have a document readable. Also, if the CAP system interfaces to a phototypesetter, photographic supplies are used instead of paper. Or if the composed copy can be electronically sent to the outside printer, no supplies are used other than draft copies.

The cost per document is also reduced first by the more copy per page. Also due to the fast turn around time for updating a document, time costs to produce are dropped as is the need to keep a large inventory of finished production on hand. Lower inventory costs mean lower storage costs.

Forms are a special area that can benefit from CAP. The reduction and avoidance of keeping an inventory of blank forms is possible with CAP. Invoices, packing slips, etc. can be stored in original format on a CAP system. With an interface to the proper business system, the data fields can be filled in as the form is being printed.

Mailing fees can also be lower. If less paper can be used per mail piece without sacrificing readability, mail weight per piece is less leading to lower postage costs.

Intangible benefits are much more difficult to quantify but are there if looked for. For a sales department, being able to respond quickly to a client with a neat, attractive, and readable proposal maybe the subtle difference that wins the account. New product fliers can be reduced fast. Price lists can be updated much more quickly.

Within a company, changes in policy can be published more quickly. Changes to train material, client or employee can be done faster.

We could go on but the idea is that the company will look more professional with documents being typeset or laserset.

There is one caveat to remember. Not every written communication needs to be laserset. Letters and memos to individuals certainly need not be. The time necessary to laserset a personal letter / memo is just not justifiable.

COST JUSTIFYING

Cost justifying is never an easy task. Either one feels like a psychic by "pulling" numbers out of the air. Or one looks "forward" to the the tedium of actually measuring / tracking costs now.

A good place to start looking at justification is the above listed cost savings area under benefits. There are two other costs area that will help in justifying a CAP system, transportation and time.

Along with the cost of an outside service is the cost of transportation or postage for delivery back and forth. For a time critical document, this can be quite costly using express services. But just normal mail certainly adds to the cost and is often over looked in cost justification.

By bring composition in-house, only the cost of delivering to and from the printer is retained. Also, the chance of loss during delivery is minimized.

Time charges is another area often overlooked. The time spent reproofing is drastically reduced; as is the time spent "waiting" for an announced pickup or delivery. Both time areas are difficult to track, particularly the last. This is because the time is usually stolen from another project, lumped together, or lost due to divided concentration.

We reported before that an average Fortune 1000 company spends between 6% and 10% of gross revenues on publishing. In some companies it can go even higher. The potential for savings is great; some say as much as 50%. The only verification we have seen is with a study done by Digital. They have been able, with a combination of solutions to drastically cut the cost (more than 25%) of a constantly increasing expense.

FINAL WORDS

Now we have described CAP and DTP. DO we recommend it? Yes. Once a company tries it they will stay with it. Again the caution, once the project for implementation or investigation is approved, keep in mind the corporate environment, the datacomm, the philosophy, and who will operate it.

The needs (desires) of those interfacing to it are also important. Can your printer accept data files on diskette, magtape, over telephone lines, or not at all? Does data and text files need to be shared? Is the resource to be shared? Etcetera.

Also remember, as with any new system, there is a learning curve. In the beginning time may not be saved. Yet with time passage, the reusability of formats, and faster turn around time will become evident.

Will printer - print shops be replaced? No. Throughout this paper we have mentioned the need for final printing. Be a document laserset or typeset, large volume printing will be done on print presses. This, for most companies, will require an outside print shop.

Nor will typographer or layout artists disappear. Their job functions will change. They will, and some have, begin to use the new technology.

We do think, if you try it you will like it.

BIBLIOGRAPHY

1. ELECTRONIC PUBLISHING SYSTEMS, Datapro Research Corporation, 1986-1987.

2. Corporate Publishing, InterConsult Inc., 1987.

3. "Desktop Publishing", Keith Cutter, INTERACT, November 1987.

4. Electronic Publishing - A Guide to Assessing Your Needs, Compugraphic Corporation, 1987.

THE FALL '88 MIGRATION: NEW DIRECTIONS?

Presented by: Charles H. Finley, Jr., ConAm Corporation

"DEC has it now" or so the ads read. What does DEC have now? Is it a good thing to have? Do we want it? What will it cost us? Can we afford it? Being the curious sort I set out to learn more. Upon further examination I discovered, much to my relief, that what DEC has is not some new dreaded disease, but a broad, compatible line of products all running under the same operating system, with some inexpensive, low capacity machines at the low end, and some very expensive but very large machines at the high end. What they also have, now, is a solid networking and data communications product line. Moreover, they are working hard to successfully integrate both IBM compatible and MacIntosh compatible products with their larger computers. They probably also believe in motherhood, like apple pie and a few of them, I'm told, listen to Willie Nelson from time-to-time.

The HP 3000 we know is a family of computers that share the same or very similar multi-user, multi-tasking operating systems, MPE and MPE XL. At the low end of the HP family there is a computer suitable for use with only one terminal and priced such that many companies, organizations and individuals consider it cost-effective. At the high end, there is a large, powerful computer that we understand is able to handle as many as 200 users concurrently. Moreover, through networking and data communications, multiple computers can be linked to form an even more comprehensive distributed solution to data processing problems. In addition, personal computers can be networked with the HP 3000, thereby allowing the user to derive benefits from the PC world as well as the HP 3000 world. HP has also made products available to allow HP 3000 connections to IBM, DEC and other manufacturers' computers who use their proprietary communications protocols and networking disciplines. HP has also been one of the staunchest supporters of international standards in networking and data communications. They have offerings in the X.25 arena as well as the Open Systems Interconnection (OSI).

Digital, we believe is in a similar position to HP both with respect to the range of offerings on their VAX computer family and in the range of their data communications offerings. A notable difference between HP and DEC, however, is at the high end of their product offerings. DEC has for a long time offered larger VAX computers than the largest available HP 3000. Also, the VAX cluster and the multi-processor offerings allow

for an environment that supports more transactions per hour at the high end than does the HP 3000.
HP we know and DEC we believe both have a family of highly compatible computers that allow the same software to be used on the largest machine that runs on the smallest computer. Few people have the illusion that this is true of IBM's offerings. IBM offers several disjointed computer systems that are only barely compatible with each other. To add to the confusion the power of these different computer families overlap.

Most of you probably already realize that it is becoming less and less useful to talk about only the largest computer we own when we discuss alternatives for getting all of our work done. Rather than a computer system nowadays, it is probably more appropriate to speak of a "computing climate," the "climate" being a collection of data processing equipment consisting of at least one mid-range to large on-line transaction processor (HP 3000), and one to many personal computers (PC's) integrated together (networked). This commercial data processing climate is what HP, DEC and IBM all offer to various degrees.

As the number of choices in computer products increases, I believe that users will tend to select one or two primary vendors and assemble their data processing climate from these. The number of alternative computer and networking products seem to be increasing exponentially and this can be overwhelming to a user. It is possible that in a few years, if all alternatives are to be considered before some new application is implemented, an MIS staff will consist only of evaluators. This is not reasonable given that other problems need to be solved as well.

Given this belief that vendors who are able to provide an entire "climate" are the ones who will prevail and prosper in the long run, this paper will consider only the migration of users between HP and DEC or IBM. Other changes are taking place but the author believes those considered here are the most significant to the HP 3000 user.

In a paper presented in the Spring of 1988 the author reported that most defections from HP to IBM and DEC occurred due to the need on the part of the end-user for more transaction processing power. It was also found that most users who had changed would have preferred not to have changed.

By the time this talk occurs, some HP Precision Architecture machines will have been in use for several months. Also, we will have moved closer to the date when the 955 (or some other larger machine yet unannounced as of this writing) will be available. The author will continue to interview users during the months and weeks immediately prior to the conference and will be prepared to discuss how things have changed since the Spring. The talk will explore what migrations have been taking place most recently, and will also address such questions as: Are HP 3000 users moving to other manufacturers' systems? Have some who thought they might change decided to stay with HP? Did any HP users start to leave and change their mind or leave and come back?

This session is expected to be more of a discussion rather than a presentation. Audience participation will be not only welcome but encouraged.

CASE - A Way out of the Software Trap

Geoff Davies

RAET Software Products BV

The Netherlands

1. Introduction

One of the hottest topics in the computer industry today is CASE, which stands for Computer Aided Software Engineering (or Computer Assisted Systems Engineering). Is this another meaningless acronym you see for a while, that disappears in a short time, when the excitement dies down? The acronym could disappear, but the concept will certainly not go away, because design and construction of business application systems is so vital a dimension of commerce and industry today, that our "profession" finds itself under increasing pressure to behave as Engineers.

Engineers today are assisted by computer technology in the design, visualisation, manufacture, testing, service, and quality control of the products of their discipline, such as bridges, buildings, automobiles, weapons, satellites, computer chips, and so on. The terms CAD/CAM (Computer Aided Design / Computer Aided Manufacturing) and CIM (Computer Integrated Manufacturing) are widely known.

So it will go with Application Software development. CASE is the term rapidly gaining acceptance for the automation by computer of the Software Development process. Automation is being hailed by many as a way out of the **Software Trap** - the trap we find ourselves in when we can't make vital new systems quickly or well enough, because of the burden of holding together inadequate systems developed in the past.

This paper will explore the background to CASE, the benefits it can deliver, and suggest an ideal toolset, especially with regard to the HP3000. Finally, some industry trends are examined for a view of where CASE is headed.

2. MIS Quality and Service goals under pressure

The march of technology in our business has not diminished in any way the pressure on MIS to improve its performance in the delivery of support to the business activities of our corporate masters. Everyone today is aware of the incredible price-performance gains in the computer, with a microcomputer being almost as common a feature of the middle class household as the television set.

Corporate leaders are therefore asking themselves why they do not perceive similar advances in the delivery of quality support systems for business activities. It's becoming such a glaring deficiency that general business publications are examining the problem, and discussing the effects and possible remedies. Our dirty linen is being washed in public.

Are we doing such a bad job? Corporate analysts say that, on the whole, we must be able to do better.

Data Processing was once a black tower, where magicians wove their secret spells, and spoke in strange language that awed and mystified the tremulous user. The minions of this domain were known to be fickle and some even had a reputation for blackmail - the secret knowledge possessed by them would leave their employers paralysed if they left the enclave for pastures new.

And indeed, little has changed in that respect. Today, experienced programmer/analysts, operators, and management are as scarce as ever - which is good news if you are one of these.

In our efforts to keep up with demand, and to maintain quality in the delivery, support and maintenance of business systems, we have become a serious drain on corporate finances, yet seem to get no closer to answering fundamental questions about the service we provide. Why do development projects so often run behind schedule and over budget? Why can we not repair software defects as quickly and easily as an engineer can correct hardware problems? Why must important enhancements to critical business systems, necessary for competitive advantage, wait so long to be scheduled and implemented?

It's generally agreed that the reasons for this, at least in the area of business applications, are to do with the ability of MIS to take the expression of a business problem and solve it with a computer-based solution. Among the reasons most commonly cited:

User to MIS communications is poor

MIS understanding of users' needs is poor

Users keep changing their requirements, increasing the maintenance load

Good programmers are scarce

MIS ability to plan and estimate is poor

MIS development productivity is low

MIS quality control is almost non-existent

There are many other possible explanations (excuses?), but the overriding impression is that MIS, who are supposed to be able to provide the total service for users, simply are not sufficiently professional in the delivery of their service.

User to MIS communications: shouldn't the responsibility rest with MIS to behave as business analysts and CLARIFY a user's requirements before a single line of code is written?

Scarcity of technical resources: where is the real problem - is it programming or analysis?

MIS planning and estimating ability: other departments (engineering, manufacturing, distribution, etc) can submit well-managed business and project plans - why can't MIS?

MIS development productivity: little gain has been made in development productivity in the USA in recent years. Few can even measure it, in fact, system development is widely regarded as the last uncontrolled business activity, and one for which few benchmark measurements exist.

MIS quality control: contrast the quality control procedures (if any) of the MIS department with those of manufacturing, and there is a yawning gap. And that should come as no surprise - QC in engineering departments is based on a rigorous discipline stemming from the recognition that, ultimately, customer satisfaction (and safety) will determine the success of the corporation. MIS has only internal customers, we have an informal relationship with our customers, and they have only one choice of supplier. But MIS can build systems, on which the business must depend to survive, with no QC rules - and those systems can be grown and extended over decades to massive networks of thousands of terminals and enormous transaction volumes.

So I would suggest that we, as MIS professionals, whatever the size or budget of our department, should decide firmly on a strategy to upgrade the service that we offer, and aggressively implement that strategy while we still have a choice in the matter.

3. Strategies for MIS to reach Quality and Service objectives

Very briefly, we will look at some steps that MIS could take to approach the objectives of Quality and Service that are desirable if we are to be seen as an asset to our employers, including CASE methodologies and tools.

There are many different ways of dividing up the components of a development project, and of course most MIS departments have several projects under way simultaneously. The following four stages are a simple model, and we'll assume that it's already been decided to proceed with a project.

Design: Visualising the finished application, analysing the data and flow of data, and setting out the programming and data management requirements. Normally a pen-and-paper job, done by an analyst.

Program: Actually creating and editing programs in the selected language, and submitting a succession of revisions to a compiler or interpreter, until each program is finished.

Test and document: Iterative procedure of verifying that the programs work, going back to programming to make corrections, and finally establishing on paper or in-code how it all works and what it means (for MIS and users).

Maintenance: Does it ever end? The amount of maintenance depends on how good a job you did in the first place, in terms of NEED for maintenance, and EASE of it.

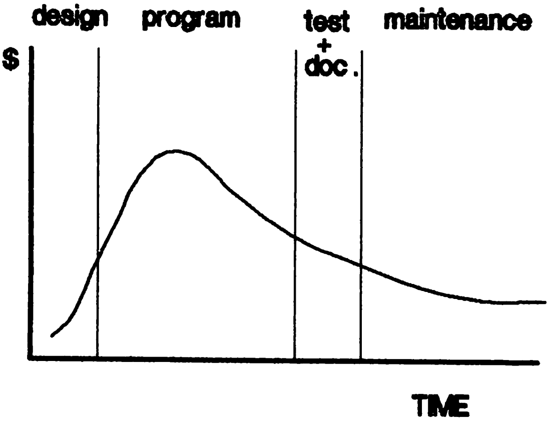


Figure 1: Traditional Approach

Figure 1 shows a theoretical model of a development project using traditional approach and the four stages (although there is no sharp line between them). The area under the curve would represent the total investment made in human and other resources in the project.

This graph emphasises that traditionally little resource in total goes into design. Why is this? Because there is little that one could do to significantly refine a design to make it more useful, once you have a few flowcharts, a list of data items, and some screen and report layouts. "Let's get on with the programming", and of course we tend to design as we go. So the bulk of cost is indeed in programming.

Testing and documentation are perhaps given MORE weight than is real in the traditional DP world. It tends to be a haphazard activity, the programmer tests his own programs, and documentation follows later - after all, we don't like to write pages of notes if we think the user might reject the system until more changes are made! Of course, if we got the design at all wrong up front, we could be a very long time putting it right later.

And maintenance, where 80% of America's programmers are busy (as Wall Street Journal would have it), just never ends. Unless the documentation was comprehensive (even through earlier maintenance), we have a lot of code to read.

What we would like to accomplish is to get the curve flatter (it makes scheduling easier), lower (it reduces costs), and for the delivery point to be nearer to the start point. There are some ways to help this.

4GL

A fourth generation language can reduce the programming load. 4GL's boost productivity enormously in low transaction volume applications, and also are very effective in rapid prototyping. They have a great ability with data manipulation, and for ad hoc report and inquiry applications. The penalty is paid in performance, and in transactions with any complexity of data management.

RelationalDatabase

Information-retrieval-intensive applications benefit from relational data management, and the associated retrieval language is usually easier to code with than Cobol. Again, the penalty is paid in high-volume applications, and you still need traditional or fourth generation language for full-function application development.

Code analysers and restructuring

There is substantial growth in this segment - the advantage comes in making old code maintainable. Obscure algorithms developed twenty years previously can be made readable for today's analyst.

DataDictionary

Implementing a standard data dictionary is a very real way to introduce some productivity - making data definitions re-usable by programmers on a team, or maintenance programmers in the future, reduces the amount of redundant coding considerably.

Generators

Code generators and report generators are a good way to re-use programming done by somebody else. Usually driven by a procedural language, you can get skeleton or even complete programs from a few statements.

The problem with all of these, and the many other productivity tools available today, is that few are integrated, and there remains an almost obsessive emphasis on the **program** as an object of attention. It seems to be overlooked that the objective is to build **application systems**, that's what the business needs us for.

Furthermore, how do we measure that we are in fact gaining in productivity at all, and to a sufficient degree that we can assert that our service is improving? Development productivity, as mentioned earlier, is an almost immeasurable quantity. The most commonly cited measurement is "lines of code per day".

Two problems with this: first, what constitutes a line of code? A line of Cobol? A field defined on a screen? A line of a DBSCHEMA? Is a replaced or deleted line a line of code for productivity measures? What if I copy 1000 lines from another program, for a "same as except" purpose?

Second, if a line of Powerhouse code can do what ten lines of Cobol do - am I ten times as productive? Is this true if the other tasks surrounding the programming (design, testing, problem resolution, editing) take the same amount of time anyway?

Understanding the productivity average for your development center is important if you are to be able to truly know that you have made improvements. Knowing productivity by **function** and by individual **personnel** can be very helpful in determining what resources to apply in a development task. How useful would it be if you **knew** the average time it took a skilled analyst to produce a transaction of medium complexity, when estimating time and cost for a new project? And the time it takes a trainee programmer to produce a new screen display for an existing system, including testing and documentation update?

Your manufacturing department has this sort of information, relevant to their operation.

In all of the tools available to the HP3000 user today, there are very few that help with design. You can obtain PC-based applications that help you understand the data and flow of data in business systems, many even produce diagrams to use as a starting point for programming.

And yet, it is in failing to get the design **right** at the very beginning that our problems begin. A business application system, including all of its screens and menus, all its reports, all its transactions, data management activity and system management, form a critical structure, supporting the corporate activities. Compare it with, for example, your head office building.

If we erected buildings the way we put application systems together, we would start from a sketch plan, hand-craft the building from the roof down, every room would be a different size, shape, height, we would make all the fittings ourselves instead of using standard ones, express surprise when the owner said they wanted 12 floors, not 3, and finally we would hand it over and say "use it for a while, and tell me what you think - if it's not quite right I can make some small adjustments!" When we thought it was all finished, we would get around to drawing up the "real" plans - if we hadn't started another project. Maintaining our building would entail going into the building and rearranging it until it "felt" right.

Now let's use CAD/CAM as an example. Who can deny the value this has been to the engineer, who can now construct at a workstation a complete specification for a machine, inspect and adjust it, before having manufacturing put it into production. He doesn't tell the computer every item of detail of a gear wheel, for example. He tells it he wants a "32 tooth spur gear of radius 4.25 inches" - and then adjusts it and moves it around. Engineers quickly became familiar with their new tools in spite of resistance by MIS (who I have seen challenge the competency of engineers to select and use them).

To conclude this part, let's look at our graph again (Figure 2) and see how it might look with an **engineering** approach to development.

We have a flatter curve, with the earlier delivery of the system. The programming phase is compressed, because in an engineering approach, rather than telling the computer how to do everything (programming) we concentrate on telling it what we want accomplished. We then leave it to the computer to assemble as much of the design into our desired executable application system as possible. We give the computer the task of coding the solution, using highly re-usable code structures.

How do we implement design on an HP3000? We have to select tools that will allow us to interact with our HP 3000 terminal as a computer aided design workstation, and which will interact with all the other parts of the development cycle, giving us a complete "software factory". This is the objective of CASE.

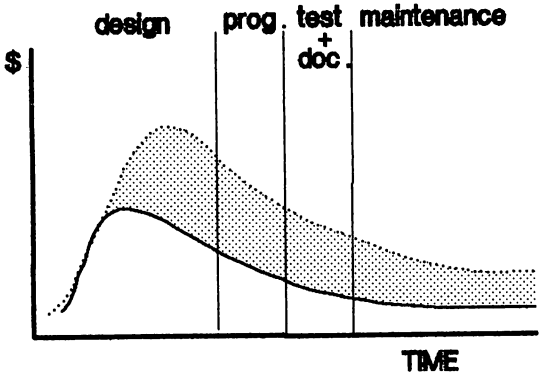


Figure 2: CASE Approach

4. An integrated CASE toolset

In this section, you're asked to forget about writing **programs** to deliver business applications. Thinking **programs** makes you relate to CASE at a level of **lines of code**, and to do so is to be fettered by tradition.

Instead think in terms of the **systems** you need to produce, and of those systems as made up of components, and sub-assemblies, rather as a manufactured product might be.

Our proposed CASE toolset may not correspond exactly to how you might perceive the vital parts of an integrated software engineering environment. There are many different ways of representing a CASE toolset, this is just one. The toolset you see here (figure 3) is oriented to a total application development environment.

In this diagram, the CASE tools are visualised in a "case" - and tidily packed away. This has no bearing on the order you might use them in. It helps you to see the interaction between each component tool.

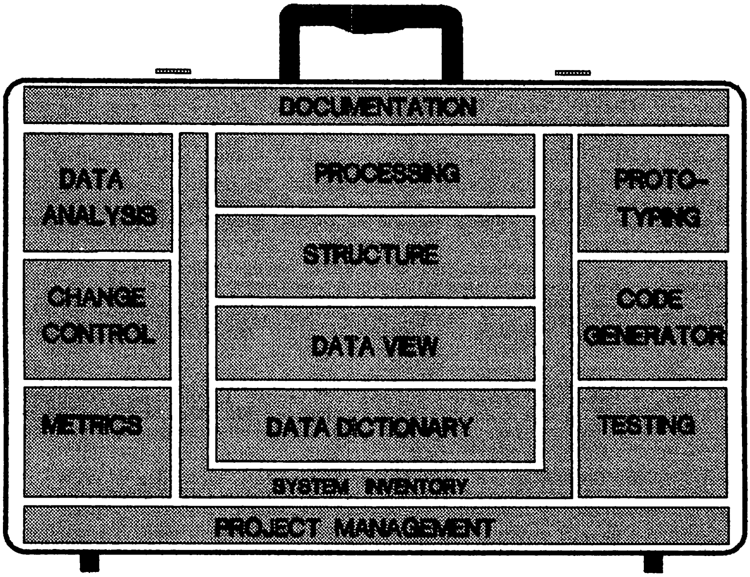

DOCUMENTATION
DATA ANALYSIS
PROCESSING
PROTO-TYPING
STRUCTURE
CHANGE CONTROL
DATA VIEW
CODE GENERATOR
METRICS
DATA DICTIONARY
TESTING
SYSTEM INVENTORY
PROJECT MANAGEMENT

Figure 3: CASE Toolset

DATA ANALYSIS

Sometimes called "front-end" CASE tools, these have been around for some time. Used to analyse the behaviour of data in an existing or planned business application, they help the development professional's understanding of the future design and data management requirements.

Large installations, especially mainframe users, tend to favour these products more, perhaps because they seem to fit very well to relational data management.

The usual approach is for a skilled analyst to survey the user's application area, capture into the PC the data entities identified, and as much information as is known about these entities. Progressively, the whole story is built up, including the logical organization of the data, the updating points, the relationship to other entities, the properties of the entities (editing, type, size, and so on).

Output from these tools is typically presented in graphical form, in **entity-relationship diagrams**, and **data flow diagrams**. Some tools are directed at specific structured programming techniques, and produce diagrams in a compatible form.

Increasingly, use is being made of the output directly, by accepting the entity-relationship information as an initial data model for the Data Dictionary, especially on mainframes.

CHANGE CONTROL

Whether it's the introduction of a completely new application, or the adjustment of an existing system, Change Control (or Change Management) is a vital aspect of controlling a lively development environment.

Volumes have been written about effective control of change and its impact on existing Management Information Systems. Micro and minicomputer users are notorious for their cavalier attitude to changes in a running application. Mainframe users have been in the business long enough, and in a hitherto more complex environment, to know that these systems are fragile things. Quality Control is closely linked - although we don't tend to think of that.

Most old hands have had the experience of a "minor change" causing massive disruption to a critical system. It's all too tempting to think we know all we need to, and make a "quick fix" that later turns out to have an effect we did not expect. The problem is exacerbated with interpretive code environments - we often want to try it and see in a live usage. Even when a backup is available to undo the damage, the disruption can be fearful.

So a management information system for MIS itself is needed - whereby procedures are in place to control changes to sensitive systems. To return to the manufacturing analogy - change to established design normally goes through an EC (Engineering Change) approval process, with QC, Engineering, Marketing and Shop Floor inspecting the change and commenting on or planning for the effect of this.

Our ideal Change Control system will include forms for submitting problem reports and change requests; a submission and registration procedure; a design review, comment and approval procedure; quality assurance procedure (by which sign-off is given that impact analysis, check on relational integrity within programs,testing, documentation, and user-advice have all been done); and a release procedure. These procedures can be real-time and continuous, in a DP shop there does not even have to be any paper. Why not give a change/problem report facility entry point to users on all application systems?

A further issue arises here, and that is controlling the versions of software in use, especially in a distributed systems environment. Change control should therefore take account of the version in use, at the point where change is needed, and being aware of the effect on other current or future versions.

By dealing with the very natural process of change in a professional and **engineering** mode, we enhance the quality of our service, and gain greater confidence from our clients.

METRICS

Repeating the earlier assertion that productivity of the development center is virtually impossible to measure objectively, we should look for tools that will help with this, if our development resources are substantial.

There are many HP 3000 users in the world with multiple development centers, with many development staff at each site.

Our ideal integrated CASE toolset will permit us to **measure** our production and productivity. This is very important, if we want to collect project progress information, but we will come to that later.

Here we raise a concept that has not yet been mentioned - **"Object Oriented Programming"**. This concept is emerging in a PC development environment - we see an object-oriented interface in the HP-NewWave environment, and the famed Macintosh interface. At the start of this section you were asked to forget about lines of code, and programs, but to think in terms of systems.

Object-oriented **development** is more encompassing than **programming** but the idea is the same. Development activity and progress is easier to express in user-oriented terms, to the users themselves, why not measure in the same terms?

Our CASE tools will help us develop **"objects"** such as screens, reports, transactions, menus, database definitions, and so on. Let's measure our progress using those terms. Integrating metrics to object development means that our Metrics module logs resources expended in accomplishing the development of a unit. For example, let's take a screen as an object of development.

Metrics logs for the screen, the total sign-on time (programmer resources), CPU cycles for design, CPU cycles for forms generation, and also can study the make-up of the screen to arrive at a complexity measurement. If you want, it can also identify who did the job. Now, this is not such a horrifying idea - in most other disciplines these measurements are indeed done. Comparative data on the performance of individuals may be more draconian than we could bear, but of **classes** of personnel (trainee, advanced, expert) may be very useful.

Commercial software specialists who do development work by contract would find this very useful as a basis for charging, as indeed would internal corporate development centers who would then have a practical basis for cost distribution to client departments.

SYSTEM INVENTORY

Now we come to the heart of our toolset. As with a manufacturing product data management system, our CASE tools will allow us to keep **all** system specifications in **one place**. where the developer(s) can all access it uniformly. In terms of Data Dictionary, this is not such a novel concept - but for the other items, it's quite uncommon up to now.

We will look at each of the four categories of information retained in this repository shortly. The System Inventory can only be useful if it is accessible interactively for **system designers** (note that we don't say **programmers**).

Adding, editing and manipulating specifications requires a very active interface commonly referred to as the **Designers' Workbench**. A day in the life of an analyst or programmer changes radically when using such a tool. You spend little or no time using full-screen character-mode editors; the Workbench presents your SI to you in a formatted and organised way. As soon as an item is added to the inventory, it's available for another designer to use.

A good workbench will provide standard objects, from which you can derive further standards of your own (such as standard screen and report layouts). Objects can thus "breed", through "same-as-except" derivations, and the components of a target application system can come together very quickly.

Progressively, as you describe your design, you can refine and enhance it. Alteration of specifications organised in this way is light-years ahead of reading source code to locate where changes must be made. Because our tool has a very active inventory, the effect of changes proposed can be detected and propagated quickly.

Documentation of design, long a despised task of programmers, becomes a task of attaching an annotation to our "objects" describing only what makes it different from any other similar object.

The SI thus forms a complete design specification for the target Application, or in manufacturing terms, a **"Bill of Materials"**. And like any manufacturing product data management system, you can obtain very useful information to aid decision-making: where is this data field used? what if I extend its length? A bill of materials can be "exploded" to show down to the lowest level all components and sub-assemblies.

Productivity gains in analysis of change effects and in re-use of standard objects are quantum. Factors of 10 to 20 in this area are not uncommon.

A designer familiar with the System Inventory becomes concerned only with differences in objects - designs produced by others are thus accessible and transparent, making maintenance a process that is not as arduous as poring over listings. Breaking a problem application down to detect a fault requires no program libraries, no UDC listings, no programmers' notes - you break it open in the System Inventory, and examine the component objects, self-contained pieces of the design.

Because the whole design is available to all design staff, division of work to specialists becomes a simple matter, for example into screens, reports, processing, structure. Measurement of progress (even in the absence of Metrics) is facilitated - counting components at least gives an objective measure. Try coming up with a simple and repeatable way to estimate the percent complete of a source-code program!

And of great importance, "programming style" becomes minimised in impact. The "software Picassos" among us may be perturbed by that - we believe that our own style is outstanding, and want to leave our signature on our works of art - but we can't stand maintaining code of others, because they are never as competent as ourselves. The word "elegance" turns up frequently when programmers are describing arcane coding problems.

We will now examine the contents of our System Inventory, and how it helps us produce better systems faster.

DATA DICTIONARY

This is a repository of "Data about Data". Data Dictionaries have been around for some time, most vendors offer one, HP 3000 users even have a choice.

The Data Dictionary keeps all we need to know about data in an accessible location, and all developers use the DD to reduce redundancy and error in their use of data items. The information kept here is **fundamental** or describing the properties of data items, **organisational** defining relationships, and **physical** describing for example disk data management.

In a CASE environment, where re-usability is a major objective, and where the properties of a data item are part of the "object definition", we need to know a whole lot more.

Added to the usual descriptive information in the DD, such as **Identity (name), type, and length**, etc, we also want to define other properties, that will be available to our application. So we also want to know **headings** for use in reports and displays (a one-character code with a long name might justify a short heading); **security** or ways to identify create/update/read access; **ranges** for automatic input validation; **tables** again for input validation; **editing** for input and output; **locking** if required to prevent simultaneous update by two transactions; **defaults** when not filled-in; **HELP** to display at input if the user is uncertain; **structure** if an entity is part of another entity; **sub-fields** if an entity has them.

This is not an exhaustive list. The important point is that in our ideal toolset, all of these are properties of the data item, and you do not have to code them to have them available in your application. If the data item is accessed, all its properties are automatically there.

Because defined data about data is all in the inventory, useful information can be given online or off-line to the designer. Decisions Support is given by cross referencing; where-used; search and retrieve. Defining new data objects is rapid with "same-as-except" activity.

Maintenance happens faster and with better results - because the designer making a data change can inspect the ripple effect of that change. Much maintenance of finished applications can be carried out simply by selecting and modifying a simple data entity definition (for example ranges, editing, prompting, HELP).

The ultimate beneficiary is the application user - data items are presented, prompted and handled more as he intended when he first explained to the designer of the existence of the items. He gets consistent treatment from his application - because editing, validation, prompting, annotation are always the same, rather than a different programmer's interpretation each time.

DATA VIEW

Defining the Data View is the process of telling your CASE toolset how the User wants the application to **look**, and to some extent, **feel**. From the on-line viewpoint, this is the layout of Screens, the way exceptions and errors are handled online, the nature of user prompting, and menus.

For off-line operations, the layout of reports is part of the data view.

And navigating through it all is also part of Data View. What's different from traditional programming is that we store standard and special Data Views in the System Inventory, as **objects** for managing and maintaining. All the facilities of the Data Dictionary are available for constructing the Data View - headings, edits, standard data validation, and so on. And because we've defined data entity relationships, associated data items can be mapped into screens and reports together (i.e. groups of elements are presented together for input/output).

Now we should also attach to our Data View objects some other information, mainly a reference to special processing, if any (such as related-item validations, for example, three input fields might have to add up to 100), and, of course, HELP describing how and what to do at a menu or screen. This approach separates **screen** processing from the driving transaction - and again makes maintenance easier by breaking the overall application into manageable "chunks".

You should have tools in your Workbench with which to define and edit Data Views, rapidly re-use existing components (same-as-except) and with which to inspect and adjust the actual appearance of it (e.g. screen painting). You should never have to enter code-inspection.

What are the benefits of separately defining Data View? Because it's here that 80% of user acceptance problems occur, then by reviewing your design with the user, BEFORE the tougher part of design begins (processing), you can adjust the design, even before the user's very eyes. It is even worth considering giving the user limited access to this part of the design to make the minor adjustments or type in the online guidance themselves (HELP text, etc).

STRUCTURE

The advantage of CASE's object-oriented approach is the separation of the desired application into components which are easier units to manage and maintain than programs. Systems built in traditional programming methods rarely have a clear architecture or **structure**, with the consequence that a great deal of effort goes into laboriously writing code to facilitate navigation (menus, related transactions, etc); and to handle the interaction of data management (database and files), manipulation, and data presentation (or Data View). What is odd about this is that almost all of this has been done before, but only sometimes do we take the trouble to establish re-usable code to make it easier the next time.

Enter CASE, in which a catalog of standard structures for all standard programming logic is already available, eliminating the need to program it. This does not mean that you now have to manipulate Cobol with the aid of copy-books.

CASE gives you, once again, **objects** which are pre-programmed structures for System Management (menu tree structures, for example), applications (where the component transactions are defined), and transactions themselves (online and batch). These structures can be thought of as **models** that you select, manipulate and customise through parameter settings.

Within each **type** of structure, all the management is provided automatically by model in use. Take for example an online "browse" through historical records. We select a "BROWSE" transaction model, and proceed to specify the customisation necessary to make it unique for our user. This would involve naming the Data View(s) attached, and the Data Management required (data base(s) and/or files); naming the **processing** objects (next section) to be invoked at the **sockets** in the transaction; and the relationships between the transaction and the application, and other transactions.

What you get automatically is management of database activity, function key recognition and action, correct entry point and initialisation, correct housework at end, management of appropriate HELP to the right point at the right time (carried in from DD and DV), and sensible error handling.

It's akin to selecting standard foundations, columns, bearers, partitions and roofing when designing a house.

Now we benefit in design and maintenance by isolating system navigation and transaction logic problems or changes to parameters, rather than having to handle source code and Job Control Language. Our user is happier, because getting into, around, and out of a system is **always** the same, regardless of which system it happens to be. Peculiarities and quirks can are sometimes amusing, often downright annoying, are a thing of the past.

PROCESSING

Processing, or calculation and data manipulation logic, can also be reduced to components of a structured application suite. The component, or **"logic object"**, is a self-contained and re-usable System Inventory item, with associated properties. The nature of those properties is defined by the designer, but the properties include, besides an identity: a **description** for the original and subsequent designers of purpose and technique employed; a definition of the **error handling** (e.g. display message, re-set work areas); the **work area** or **common areas** that the logic accesses; **linkages** to other processing logic objects, depending on success or failure; and other possible properties.

Of course, the actual data manipulation and calculation logic itself is also a property of the object.

In any CASE environment, it is almost impossible to eliminate the need for a high-level or macro definition language. Our ideal CASE toolset includes an interactive editor to allow you to enter new logic, or copy other logic (same-as-except) as a starting point. The editor must include features to ensure that the logic is syntactically correct, and that we work with real data (defined in the DD).

The CASE methodology relieves much of the tedious coding chores from the designer, through its Data Dictionary, Data View and Structure facilities. So it's probable that 80% of the design coding effort will be expended in this part. Our CASE Process Definition Language must therefore provide a very high level of macro capability, reducing to a few keystrokes what normally requires a substantial coding effort.

The highest-skilled analysts in your installation can preserve in a re-usable form their skill for access by less experienced personnel. Because our CASE processing puts a "fence" around a piece of logic, it becomes simpler to read it, understand it, use it and learn from it,

Again, this separation makes maintenance far simpler. You do not have to be the originator, nor have any written manuals present, to be able to isolate a problem piece of logic and fix it. And in development, the accomplishment of a logic object or process definition means that you need never handle that code again,

Contrast this with traditional programming, in third or fourth generation language. Construction of large and complex programs tends to be an iterative cycle of edit, compile until clean, add more complexity. Even parts of a program that are running OK get compiled again and again. Sometimes, the subsequent re-editing messes with code that was running satisfactorily.

Once again, it's pointed out that the chief beneficiary in the end is the user - who gets a better response from DP for maintenance and new system development.

PROTOTYPING

We now leave the System Inventory, at least in regard to changing its contents. But Prototyping, a proven technique employed by many for verifying design, can participate in the SI as well in our ideal CASE environment.
If we did not get the **look** of our system right using the Data View facilities, we can generate a prototype application and actually **run** the application. When does a prototype stop being one? The answer is probably, when you feel like the design is nearly finished.

To truly prototype what the finished application will look and feel like, and picture what a day in the life of the user will entail, you have to be able to go from transaction to transaction as the user would, not by starting Data Entry under Formspec. The user needs to see, and you also, what the screens and reports look like when there is data appearing in them.

In our CASE environment, therefore, you can check the look in the Data View parts of the workbench, as well as the prototype. But for the **feel** you need to start the application from the Operating System, as in "real life".

Which is why, the prototype is best made as the word intended - not in some special make-believe environment (such as by using dBase), but is in fact the **first effort** from the toolset, and in fact is the first reviewable version from the code generator (next).

As with verification of the look in Data View, up to 80% of rework can be eliminated from the post-delivery phase if you do this phase right. And that makes us all happier.

CODE GENERATOR

Ideally, you should use the code generator to make the prototype, because then it is a **true** prototype.

Why a code generator? Its not the only choice, there are quite a few CASE environments today integrated to a 4GL. But 4GL's often give real performance problems in highly transaction-intensive systems. A code generator that makes compilable code gives you some fringe benefits.

First, if generated code is recognised third generation language, such as Cobol or Fortran, then you maintain independence of development environment, your 3GL code is still maintainable; second, distributing applications based on a 4GL inevitably leads to high charges for runtime systems; third, distribution of compiled 3GL code gives a large measure of protection against copying of proprietary source and reduces size of application libraries on smaller disk systems.

A fundamental assumption here is that, because the CASE tools are integrated, the Code Generator can read the specifications from the SI.

The Code Generator makes more than just executable code - it should also create source code, data management schemas, initialise databases, build screen forms (e.g. VIEW), write the linking job control, bind message and HELP text files into the whole, and act on your instructions for organization of the executable libraries (USL's etc).

A number of benefits accrue from the CASE Code Generator. Reliable and correct code generated does not have to be debugged - only SI specifications need to be; consequently, the old cycle of edit/compile over and over becomes less prevalent. CASE designers don;t feel the need that programmers do, to get one program perfect before going on to the next one. In the CASE environment, you could design for weeks and never generate or compile in all that time. Then, as you get to prototyping and refinement, you begin to need to generate.

It becomes possible, and I have seen it, that contented design staff schedule all their generate/compile activity for the back-shift. It's more conducive to good design to simply work with the interactive tools through the day, bypassing the edit/compile programmer's cycle with it's trips to the printer, sessions with spook, and "just one more small change and it will be right". Less erratic, unscheduled, and CPU-intensive activity makes for good response times.

TESTING

The automation of Testing is a late-emerging part of CASE. There are some tools available that allow you to automate an interactive terminal session that signs on, enters transactions, creates reports, deliberately makes mistakes, and so on, according to your own scripts and then gives you a report on what was different from the last time it did it.

Specification of standard test procedures like this is vitally interesting to software package developers, or to those with colossal user populations.

You could check for code that was never entered (why do we need it? Maybe our test was inadequate, or we have a logic error?), and try all the "ridiculous" values for input ("I never would have expected a user to enter minus 5 for the month").

Another interesting area we want for our ideal toolset is an extension of the Data Dictionary, to describe the intricacies of data behaviour in the finished application. This might for example describe the usual distribution of number of order detail lines per order header; the usual distribution of letters in the NAME field of a NAME & ADDRESS group.

A test data generator would then construct full data bases based on the statistical information you predict, for you to test. One of the hardest parts of testing is to get enough realistically filled records to make all screens and reports look as they would when the application has been in use some time. Or even, to be able to estimate the response time of an inquiry application accessing a data base with a million records, and a complex structure. Test data generators can do this.

DOCUMENTATION

There are already a number of tools on the market to help us do what we least like doing, documenting our systems. But they rarely are able to shine light on purpose behind logic that they scan. Our System Inventory is self documenting. The designer is disciplined more to create a small amount of descriptive information when creating Processing objects, where he is outside of the boundaries of pre-structured or special-purpose objects.

All the pre-structured and special purpose objects (screens, data elements, reports, transactions, menus) have such a defined logic and purpose that further manually produced documentation is redundant.

An active System Inventory therefore carries all its documentation within it. The CASE toolset only needs to provide access to the design for management reporting and designer review. This is accomplished by reports and inquiries, and is an application that should offer choices of levels of depth and complexity.

Calling for a full set of reports describing all structures, processing, messages and text, screens reports and data dictionary - is asking for a System Reference Manual. It's always up to date.

Calling for menus, screens, and report layouts, annotated with the validation rules, ranges and associated HELP text for all input fields and action screens - is asking for a User's Manual. And it's also always up to date.

DP staff benefit - that burden of guilt for incomplete documentation lifts from your shoulders. Your client user benefits also - machine-produced documentation is able to be customised and formatted - I've seen very smart manuals produced with corporate logos and other frills, using a laser printer and CASE documentation tools.

PROJECT MANAGEMENT

These tools are not exclusively the preserve of CASE, but when you run the rest of you development environment so well, why not underpin it with an integrated toolset that helps you maintain control over large development projects and maintenance? All project management tools include a Critical Path algorithm, defining the shortest path between the start and finish points, and many include cost control and resource management facilities.

Integrating Project Management to the rest of the toolset means measuring progress (accessing the SI) and determining if required stages have been passed yet. It also means that Metrics can be actively employed to determine, based on previous performance, when a project will be completed.

This can only lead to trust in MIS, and more satisfied users.

5. Industry Trends

We shall now review recent developments in Computer Aided Software Engineering, and see whether any trends are emerging.

Technology

CASE workstations driven by proprietary CASE software are expected to become a growth area. Analogous to CAD workstations, the CASE workstation will use **icons** to ease the selection of objects for design activity, and **windows** to permit rapid navigation through the toolset, and to run testing and design side by side. Some engineering workstation vendors are already producing CASE workstations for UNIX systems.

Workstations which are fully compatible with the target environment are today quite common and the new HP 3000 LX models could be considered to be development workstations for HP3000 corporate systems.

Start-ups

Business Week reported in May that the worldwide market for CASE tools could hit $2 billion in 1992, and in the US alone, $800 million.

It's no surprise, then, that there are many people getting into the act. In the same article: a San Francisco consultant sees two new CASE start-ups a day, and an analyst reports that he knows of some 100 CASE companies, mostly less than two years old, and most funded at more than $1 million.

Perhaps we'll see the same explosive sort of growth that the micro started.

Standards

It only becomes interesting to attempt to establish industry standards for any new technology once there is a discernible movement to embrace that technology by large numbers of pioneers. There are no standards yet for specification of software design, and no standards for productivity measurement. The IEEE has struggled for several years to come up with a "single perfect measure" of software productivity, but few believe it is possible.

What is more likely is that the new CASE technology and object-oriented design and programming will permit accurate measurement of development effectiveness, but comparison to former methodologies will not yield any accurate figures, because the old methods of measurement are so imprecise.

Alliances

Recognition by the computer industry that growth is stunted by the inability of corporate clients to take full advantage of the power of new computers, because of the maintenance and development backlog, has greatly accelerated the interest of vendors in having CASE tools available.

lowering the cost and improving the quality and service offered by MIS might make more budget available for more hardware purchases, and growth in user populations can only lead to increased peripheral and capacity purchased.

Consequently, we are seeing major hardware vendors and software companies teaming up, and manufacturers of discrete CASE tools getting together. Even major corporate computer users are invited to have their say.

In 1985, fourteen leading aerospace and defence contractors formed a limited partnership called the Software Productivity Consortium. They have an invited group of CASE vendors called the Guest Systems Council, and jointly they are attempting to formulate complementary strategies for their future mutual benefit. The last press statement I saw was optimistic, but nothing concrete had emerged.

Education

MIS is traditionally adverse to risk, and as a consequence is not yet ready to embrace the new technology. Few companies want to be pioneers, and few in corporate MIS want to lead the way for their colleagues. As Computerworld put it, we need a "hero in the programmer's shop".

However, a growing number of CASE evangelists such as T. Capers Jones, James Martin and David Yourdon are defining the CASE environment thoroughly, and some excellent publications from some of these authors are available. There will probably be an increase in the exposure of computer science students to CASE methodologies.

Cost

There is a huge variance in the price of CASE tools, just as with 4GL's and databases, depending on whether you're an IBM mainframe user, or a small mini user.

A total life-cycle CASE environment recently announced for mainframe IBM by one of the "Big Five" accounting firms, includes an IBM PC based methodology front end at $50,000 for a site license; a design interface for filling the repository at $7,000 per networked micro workstation; and a generating/implementing back-end that costs a hefty $200,000 for a single license.

At the other end of the spectrum, the most popular data analysis PC based front-end is around $8,000 per copy, and a full application development and documentation environment can be had on the HP3000 for just $30,000.

The cost of the tools is, of course, offset by the gains in productivity, and the competitive advantage to the corporation of having high-quality and maintainable systems available faster.

CASE is here, and it's growing in importance. The signs are in the industry that it is going to reach all of us, very soon, and very pervasively.

---

6. Summary

We've reviewed the pressures on MIS to upgrade the quality and service which is its responsibility, and we've seen how executive attention is becoming focused on software development productivity as being at least part of the problem, where it exists.

One of the possible strategies to help us drive towards MIS quality and service objectives has been examined up close, and one ideal set of tools has been proposed. The ideal set of tools would transform our development center from a craftsmen's workshop, into a professional software engineering center, with consequent advantages already enjoyed by engineers in other disciplines.

And finally we have glimpsed a few of the developments relating to CASE in the computer industry, which augur for a healthy period of penetration and growth for this technology.

Attendees are invited to discuss the CASE approach further at the RAET Software Products exhobot, number 1015.

TAKING A SHORT BREAK ...

Michel Kohon
Tymlabs Corporation
811 Barton Springs Road
Austin, Texas 78704

"Please Do Not Interrupt."

Tacked to a door, this notice might be appropriate for a Meeting of the Board, or a poetry reading. But if it appeared on a computer, it would certainly belong on the HP3000 under MPE.

Although the architecture of the HP3000 is very well suited for handling interrupts, starting or stopping a process is a painful endeavor for MPE. The `:RUN` command costs so much that new MPE versions have an auto-allocate procedure. This may be fine for starting a program, but it is no help when switching from one program to another.

I spent a long time at a merchandising company where people work in an interrupt-driven environment. Phones ring, display monitors update commodity prices every 15 seconds, and decisions are made in a very short time. If the merchant is looking at his long/short position and a customer calls about the status of his current shipment, do you think the merchant will wait one minute while his currently-running program shuts down and a new one begins? No! He won't use the computer. But if it took only a few seconds he would.

"Process switching" is the ability to access one program while running another one, a facility which can provide both software developers and end-users with increased power and flexibility. For process switching to be useful, it must be fast, utilizing system resource in a very efficient manner. Could such a facility be implemented on the HP3000?

The MPE BREAK function provides an interrupt capability which is inexpensive in terms of system resource. This paper reports on the inner workings of BREAK, and suggests how this knowledge might fit into the overall prospect for implementing process switching on the HP3000.

BREAK: A Guided Tour

To understand precisely how BREAK works, you might begin by looking for some documentation, as we did. Will you be surprised to hear that we didn't find any documentation on BREAK in the whole MPE manual library? By a *stroke of pure* luck, however, we managed to find the section in the MPE source code itself which contained the information we needed.

My description of the BREAK function will take us through the intricacies of the MPE operating system, revealing both its sophistication and its weaknesses. Remember that MPE is multi-process oriented; be ready to follow BREAK into several subprocesses.

There are two ways to call BREAK — either by pressing the BREAK key on your terminal, or programmatically by calling the intrinsic CAUSEBREAK.

Let's begin with what happens when you hit the BREAK key on your terminal. CAUSEBREAK will come into play in step 4.

1. The terminal generates an extended spacing condition (a hardware, rather than character, condition) and the terminal driver detects the BREAK, causing the ADCC, ATP, etc. to interrupt with a BREAK status.

2. The driver process calls the procedure BREAKJOB. BREAKJOB is an uncallable, privileged procedure which organizes the user process into the BREAK state.

3. BREAKJOB first hibernates all the user's sons, then sets the user in a BREAK state by turning on a bit in the 1st word of the MPE Logical Physical Device Table (LPDT). Next, BREAKJOB calls the procedure CAUSESOFTINT, which runs on the Command Interpreter stack. CAUSESOFTINT is a general procedure in charge of all types of pseudo-interrupts.

   In order to switch processes, BREAKJOB sets on the user's process pseudo-interrupt bit in the PCB (Process Control Block) using the procedure SETPSIF. It then awakens the C.I. whose PCB offset is passed as a parameter to the AWAKE procedure. The C.I. starts executing the procedure CAUSESOFTINT, as its PCB is now marked as being in BREAK.

4. CAUSESOFTINT calls the entry point SYSBREAK located in the actual C.I. code.

   If a program had used CAUSEBREAK to break, this step would be the entry point in the BREAK circuit. CAUSEBREAK takes care of hibernating the process's sons and awakens the C.I. via a pseudo-interrupt. From here on, things happen in the same way, whether you used the BREAK key or called CAUSEBREAK.

5. But what if OPTION NOBREAK were specified in one of the active UDCs? In that case, SYSBREAK would call FCONTROL to disable BREAK, and return to CAUSESOFTINT, which would re-activate the user's process.

   BREAK is disabled when the UDC is executed, as follows: The UDCINIT procedure calls the SETSERVICE procedure, which sets the LPDT break bit to 1. This indicates that the terminal is already in BREAK (although it is not), effectively voiding any other BREAK.

   This explains why you don't see anything happening when you hit BREAK while under the influence of an OPTION NOBREAK. It also means that you cannot enable BREAK with FCONTROL when OPTION NOBREAK is in effect.

6. If there is no OPTION NOBREAK, word 32 of the C.I. PXFIXED (a MPE table built in every stack) is set to -1.

7. What if there is an ongoing read at the terminal, or the program is waiting for a read (FREAD, READ, ACCEPT, etc.)? This means the user's program is engaged in a procedure called IOMOVE which interfaces the high level intrinsic with a low level one dealing with logical devices.

   This low level procedure is ATTACHIO. ATTACHIO controls the logical transfer of data between any physical I/O device and the stack. It is one of the most complex procedures in MPE. Actually, ATTACHIO is the tip of a deep iceberg which reads from, writes to, and locks any physical device (terminal, HPIB, ADCC, ATP, etc.).

   A word to the wise: If you have to deal with ATTACHIO, make sure the logical device number is valid — unless you want MPE to grant you a System Failure 206!

   ATTACHIO is called by IOMOVE to read the logical device attached to the terminal (for example, LDEV 20). When you hit BREAK on your terminal, ATTACHIO detects it as an error in I/O, and IOMOVE unlocks the terminal file control block (ACB) to let the C.I. process run. (If it is the C.I. itself which is running, the BREAK request is voided).

8. But let's return to SYSBREAK, from step 6. The uncallable FBREAK procedure is called by SYSBREAK (on the C.I. stack) in order to set the terminal file control block (ACB) break bit. During this time, the ACB is locked, to insure that no conflicting access to the file takes place. FBREAK is running on the C.I. stack.

9. Eventually, FBREAK releases the ACB. If there is a read pending on the terminal, IOMOVE tries to lock the $STDIN ACB. But since we are now running the C.I., this attempt impedes the user's process in a low priority queue internal to the file control block vector ("long wait" state).

10. Once FBREAK is completed, the C.I. checks to make sure the calling process was a session before displaying the expected colon. If it was a job from which BREAK was called with CAUSEBREAK, the output buffer is flushed.

11. Now the C.I. is waiting for an acceptable MPE command. Most commands are permissible; however, the `:RUN` command, and others which imply `:RUN`, are not. This is understandable since `:RUN` would create a brother process rather than a son. (Don't forget that the C.I. is now active.) But wouldn't it be nice to have some kind of mechanism to run a program from within a BREAK?

12. When the C.I detects `:RESUME`, `:ABORT`, `:BYE`, or `:HELLO`, word 32 of the C.I. PXFIXED is set to zero (no longer in BREAK) and the procedure FUNBREAK is called.

13. FUNBREAK locks the terminal file control block (ACB) and sets its ABORTREAD flag to FALSE. This bit will be used by any waiting IOMOVE. An ATTACHIO is fired to tell the terminal driver to clear its BREAK DIT (Device Information Table) so that the current read (if any) begins with data that has already been input and stored in a pending buffer. The ACB is unlocked.

14. If the C.I. detects a RESUME command, it returns to the procedure CAUSESOFTINT. This brings all user's sons out of hibernation and resets the user's PCB to a no-BREAK state.

15. The user's process is no longer impeded, and the program resumes.

    If a read was in progress, IOMOVE can now lock the terminal ACB to complete that read. But before re-initiating the read, IOMOVE displays the familiar (and hard-coded!) "READ pending". (How about a beep, instead of this message printed in the middle of a formatted screen?)

This is the end of your guided tour. BREAK-time is over; your program has resumed.

As you can see, a tour of the BREAK function is almost as complex as a visit to the Hearst castle, and I cannot even be sure that my description is perfect since no documentation is available. In fact, I would be very interested in your comments if you happen to know more about what is actually going on during specific phases of BREAK, and have documentation of it.

Given what we now know about BREAK, how can it be used to implement process switching? BREAK itself doesn't provide a way to actually run a second program; it merely gives access to a subset of MPE commands, which are, for the most part, useless in a business application environment. Two steps are still required: We must seize control of the terminal at some point in the BREAK process, and, once in control, we must make it possible to run a second application.

The Riviera Interface to BREAK

As of this writing, the Tymlabs product which will take advantage of this research is not yet on the market and doesn't have a name. For now, I will use the code name Riviera (that is where many Frenchmen like to take their breaks).

In order to take control when you hit BREAK or call CAUSEBREAK, Riviera substitutes its own routine for one MPE procedure called during the BREAK process. Our research indicated that the safest place to substitute would be at the point where SYSBREAK comes into play. Remember, that was step 4, where the two ways of initiating BREAK converged. This was by no means the only possibility, and was certainly not the easiest to implement. A long series of experiments led us to this conclusion — every one of them causing some kind of system failure! SYSBREAK's advantage is that it is an entry point in the system SL, and therefore can be called as a procedure.

To substitute for SYSBREAK, we perform a procedure called "trapping": the actual SYSBREAK is renamed and our substitute SYSBREAK decides when and how to call it. We do not change MPE code, and we let MPE call our uncallable new SYSBREAK, which is located in a special segment of the system SL.

Two smaller problems are also solved with traps in the Riviera interface. In order to provide access to Riviera even when OPTION NOBREAK has been set by a UDC, we trap the procedure FCONTROL with a substitute FCONTROL. Our FCONTROL decides for itself whether to satisfy a BREAK disable request or not. And since Riviera needs to know who is logged on at a specific terminal, we trap INIJSMP, which is called by the `:HELLO` command.

As I mentioned earlier, the challenge in implementing process switching is to use very few resources. For example, we don't want a procedure to have to make a disc access in order to determine whether Riviera is running. Our only choice is to set a flag somewhere in a core-resident table. We decided to use the Job-ID field of the JMAT (Job Master Table) related to the session/job which is actually running Riviera, since this field is not critical for MPE. When Riviera starts, it stores in this field the string "$DESK" followed by the Riviera version number. When Riviera is stopped by the system manager or operator, the original Job-ID is restored. This is not necessary for MPE's purposes, but is useful for any resource accounting or security system which accesses the Job-ID at logon or at logoff.

Every time I use the phrase "Riviera is active," it means that a procedure has determined that the JMAT contains one entry with the string "$DESK". Since there is no way a session or job can log on as $DESK.GROUP.ACCOUNT, and since $DESK cannot be displayed during a SHOWJOB, our trick is both safe and invisible.

If Riviera is not active when BREAK is called, we call the original SYSBREAK. If the user has access to BREAK, he gets the colon. Otherwise he gets the usual nothing!

If Riviera is active when BREAK is called, we first check to make sure we aren't in Riviera itself. (We don't want anyone to interrupt Riviera.) Riviera disables BREAK also (by trapping FCONTROL), but under no circumstances should we allow a BREAK in Riviera.

So, assuming that Riviera is active, but we are not in Riviera itself, what happens when someone hits BREAK?

First, we determine the user's group and account with a procedure which scans the JMAT until it finds the $DESK Job-ID. This procedure then returns the group and account, so that two message files, used to communicate between a user and Riviera, are opened. (These two files, BREAK and RESUME, are built by Central Riviera when it starts operation.)

Once BREAK and RESUME are successfully opened, we send a message to the BREAK file, which has Riviera as its unique reader. We then perform the following 6 steps :

1. Call SETSERVICE to disable BREAK.

2. Wait for a message from the RESUME file.

   The RESUME file reads all messages in a non-destructive manner. Then it checks to see whether the terminal specified in the read record matches the user's terminal number. If it doesn't, we pause for 1 second and keep reading messages (if any). The pause is necessary since we are running in a linear queue and other processes must have a chance to collect their messages which may be placed in front of ours in the message file. If the specified terminal does match the user's terminal number, the message is re-read in a destructive manner, and we skip to the last step.

3. Call SETSERVICE to enable BREAK.

4. Flush the terminal I/O buffers.

5. Call FBREAK to set the terminal ACB in BREAK.

6. Call FUNBREAK to reset the terminal ACB to normal, producing the less-than-desirable "READ pending" message, which we will discard in our ATTACHIO trap.

Riviera is now in control of the terminal, or rather has allocated one of its sons to the terminal. At some point, Riviera's son sends a RESUME message to the user's process. This message specifies whether we want to go into the regular MPE BREAK or to resume.

First case: we want to call the regular BREAK:

Our SYSBREAK calls the original SYSBREAK and we are in BREAK. If we were under an OPTION NOBREAK, the original SYSBREAK would kick us out and disable BREAK. This is why our substitute SYSBREAK re-enables OPTION NOBREAK when we come back from the original SYSBREAK by calling SETSERVICE.

Second case: we want to resume:

We return from our SYSBREAK and we are back in CAUSESOFTINT, which will resume our program.

If, during our SYSBREAK processing, we find an unacceptable situation, we will call the original BREAK.

The Door is Now Open to Process Switching

That is how Riviera takes control when you hit BREAK. Now we face the final step in process switching: to run any application without any modification. Three problems became apparent in our attempts to achieve this.

The first problem we encountered concerned redirection of I/Os. Riviera can run a program using the terminal on which it is active. That is fine, except for the small detail that Riviera is running in, say, session #1, while your program (now in BREAK) is running in, say, session #3 – which means that they aren't using the same terminal. How can we redirect all I/Os of a Riviera son (say Query from HP) to the terminal in BREAK, with a minimum overhead? We accomplished this by using a memory resident table, which links a certain process number to a target terminal.

There are many tables you can use, but let me give you a piece of advice. Some tables contain reserved or seemingly empty fields; avoid them, as HP might decide to use them in the future. Use only the documented fields. And here is the trick: use them in a way which doesn't conflict with MPE. Let me give you an example: Say you want to use the LPDT (Logical Physical Device Table) to indicate a condition of the terminal. The two first bits of word zero are normally set to 1 when the terminal is used. Set them to 3 for a short while and only your program will notice and use that state without any conflict with MPE. Rather than fighting MPE, just try to use it in a very gentle way.

In redirecting I/Os, we needed to trap FREAD/FWRITE/FCONTROL, but also the primitive ATTACHIO in order to bypass MPE file system. Each one of these procedures quickly checks our memory resident table to see if the I/O needs to be redirected.

The second problem we were faced with concerned the "environment", that is, the User/Account/Group in which the program is running. Riviera might be running as MANAGER.TYM,PUB, but the terminal in BREAK is most likely using a different

User/Account/Group. In order to access the correct JCWs, File Equations, and temporary files, as well as the correct posting of both elapsed and CPU time used, we needed to change the environment of the program running on behalf of the terminal in BREAK.

JCWs, File Equations, and temporary files are in tables whose addresses are kept in another set of tables called JIT and JDT, which are job related (one entry per job/session). The addresses of these tables are kept in each process stack (in the PXGLOB, to be precise), so we replace the JIT/JDT addresses in the program stack with the addresses contained in the stack of the process in BREAK, and also the JIT/JDT session number and main pin number.

At this point we could run any program from a terminal in BREAK and it would behave as if it were launched from that terminal except for a small detail: Break and Control-Y will not work! This was the third problem we faced: Since your terminal is already in BREAK, the next time you hit BREAK or Control-Y, MPE ignores it at the driver level (remember step #2 of the BREAK description). The only available solution was to trap the BreakJob procedure.

If you are already in BREAK and you hit BREAK during a read, two things happen. First, our BreakJob procedure sets a bit in the LPDT, which cues another process to send a message to the Riviera server so that the Riviera menu can be presented to the user. Second, FREAD also detects the BREAK and the program suspends itself so that next time you select that program it will resume execution at the exact point where it stopped! If you hit BREAK (while in BREAK) during anything else but a read, we emulate a Control-Y. If your program doesn't have a Control-Y trap, we terminate the program.

Now that these last three problems are solved, it is possible to run another program while in Riviera.

The Ultimate

I have described one way to implement process switching on the HP3000. As you have seen, it is no simple affair. It must handle scattered data – and unlike IMAGE-oriented products, or backup products, or compilers, process switching deals with more than one object inside MPE.

From a user's perspective, on the other hand, process switching is a breeze: Hit BREAK, and you are in the Riviera; once there, you can do something different.

In other words, you can run an application, hit BREAK, and select another program from a menu; run that program, hit BREAK again and select another program, and so on, without ever terminating any of those programs. You may be familiar with the Macintosh "Switcher" product. This is exactly what we now have on the HP3000. For example, while you are running your online order entry system, you can quickly access HPMail from time to time to check your mail. You can also test a program with QUERY, QUEDIT, MPEX and PDQ for Quiz all at hand. By suspending existing processes rather than terminating them, not only do you avoid the overhead of termination and creation; you also avoid the overhead of opening the files or data bases that these programs are using.

The Riviera product described here has been years in the planning and implementation. I don't recommend that you try to implement process switching on this scale as part of another software application. However, implementing a more limited system based on MPE process handling features, such as child processes, message files, and MPE's Control-Y, is certainly feasible.

So there you have it: How to take a break on the HP3000, and what to do with it.

TITLE: Methods of Cost Justification for Hardware and Software Purchases

AUTHOR: E. Charles Stern

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 0176

TITLE: The Best of the New Creative Decision Making Techniques

AUTHOR: Teresa Norman

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 0177

An Introduction to Optical Disk Technology

Husni Sayed
Deborah Cobb
IEM, Inc.
P.O. Box 8915
Fort Collins, CO 80525

Introduction

Laser technology found its first popular commercial use in the music industry, storing digitized sound on CDs, or Compact Disks. CDs became popular very quickly, because the disks are so compact, and the sound quality far superior to tapes and records. Now, laser technology is being used in the computer industry as well.

Optical Disk Drives, which use lasers to store and retrieve data from optical disks, are one of the latest breakthroughs in mass storage technology. WORM (Write Once, Read Many) optical disks are currently available, and have a number of advantages over more traditional storage methods. Their most obvious power comes from their tremendous data capacity, which allows massive amounts of data to be stored very compactly. Since WORM media can only be written to once, they are best suited for storing information that does not change over time.

The "write-once" aspect of WORM media does create a few problems that must be addressed. WORM optical disks are more susceptible to surface defects than are other storage media, so special care must be taken to see that all errors are detected and corrected. Also, since a standard operating system computer-disk drive interface expects that an area can be written to more than once for directory maintenance, special software drivers are required for use with WORM disks.

Erasable optical disks are now in the works, and should be available commercially in the very near future. Erasable disks offer all the advantages of WORM media, as well as the flexibility of erasing and rewriting stored information. However, the erasable technology is still in the developmental phase: it has many problems to address and a long road of development ahead. Erasable disks, and future developments in the field, will add to the allure of optical technology.

This paper will introduce both technologies, but will proceed to describe the WORM technology in greater detail, as it is currently available.

Optical Recording Methods

Optical disk technology was "fathered" by Phillips Corporation. CD-ROM and WORM drives are now being developed by such companies as Sony, Toshiba and Ricoh, along with a host of smaller companies. Erasable disks are being developed in sizes of 5.25 and 3.5 inches, by Olympus, Sharp, Seiko-Epson and Kodak, to name a few.

WORM Disks

WORM optical disk drives use optical disks that can be written to only once, as the writing process causes permanent alteration of the medium. WORM disks are grooved, much like a phonograph record but with a much greater density of grooves. Writing is performed on the raised portion rather than in the groove itself. A laser beam is used to produce a "pit" on the raised portion of the medium. Figure 1 shows a vertical cross section (along tracks and sectors) of a WORM disk.

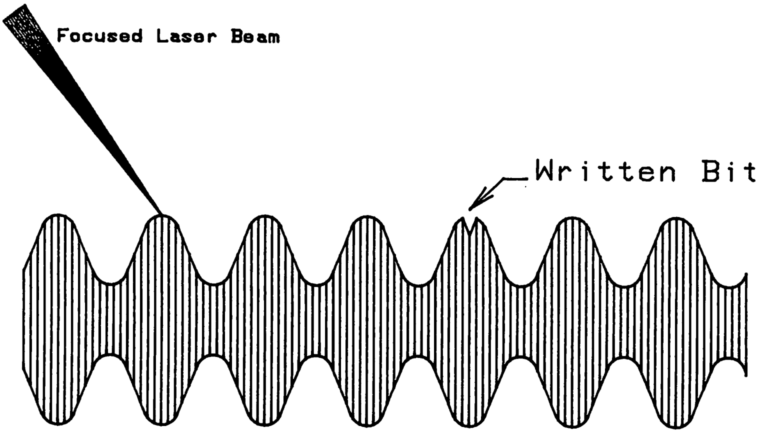


Figure 1: **WORM Optical Recording**

Each pit created on the disk represents a bit. The same laser beam that is used for writing is used to read the information that has been recorded: each pit is interpreted as a digital 1, and each "no pit" is interpreted as a digital 0. Once a pit has been created, it cannot be removed: thus information written to a WORM disk is permanent.

Erasable Disks

Optical disk drives that record on erasable disks also use lasers for reading and writing, but the technology is a bit different. On erasable disks, the recording substrate is magnetic: the value of a bit depends upon whether its orientation is "north-pole-up" (representing a 1), or "north-pole-down" (representing a 0). This is illustrated in Figure 2.

A blank erasable disk has all of its bits pointing north-pole-down (0). A magnetic coil in the drive produces a magnetic field that points north-pole-up. The strength of the magnetic field required to change the orientation of a bit varies with temperature: at room temperature, the magnetic coil is too weak to induce such a change. At temperatures above 300 degrees Fahrenheit, however, the force required to change the magnetic orientation of a bit falls to almost zero, so bits are easily "flipped" by the magnetic coil.

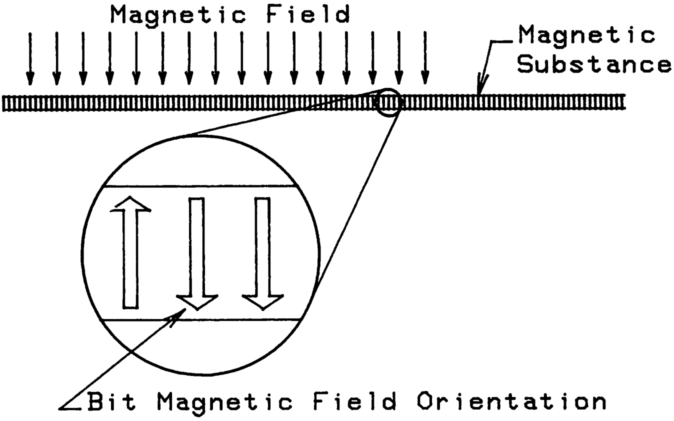


Figure 2: **Erasable Optical Recording**

To record information, a laser beam is used to heat a spot on the disk (representing a bit) to 300 degrees Fahrenheit for a few nanoseconds. In this time, the magnetic coil is able to change the orientation of the heated area to north-pole-up, recording a digital 1. Data on the disk can be erased by reversing the magnetic field of the coil (either by physically flipping the magnetic coil, or electromagnetically): with the laser on continuously, every bit that is oriented north-pole-up will be flipped to north-pole-down.

The same laser beam, at a much lower intensity, is used to read the data on the disk. When the laser beam hits the substrate and is reflected, its polarization will be rotated in either a clockwise or counter clockwise direction, depending upon the orientation of the bit being read.

Optical Advantages and Disadvantages

Advantages

WORM optical disks are rapidly becoming the medium of choice for data storage and archival, due to their numerous advantages over other data storage and archival methods. When erasable optical disks become available, the same advantages will apply. These include the relatively long archival life of optical disks, the compactness of the optical disk cartridges, high capacity, portability and ease of data access. Optical drives are also less vulnerable to problems caused by surface contamination, are not prone to head crashes, and data cannot be accidentally overwritten, or erased by magnetic fields.

Magnetic tape, for instance, has traditionally been chosen for the archival task. But magnetic tape has a relatively short life span: three years or so. Stretching or breaking of the tape and print-through (which occurs when the magnetic field from one layer of the tape migrates to an adjoining layer) contribute to the degredation of the medium, hence corrupting the stored data. Optical disks, on the other hand, have a projected archival life of 10-20 years. In addition, data access times are faster for optical disk drives which are random-access devices, than for magnetic tape which must be searched sequentially when data needs to be retrieved.

Microfilm, which has a 100 year life expectancy, is another medium commonly used for archival purposes. Microfilm's greatest disadvantage is its inconvenience: retrieving stored data is a tedious task. This characteristic makes microfilm great for storing information that is rarely (if ever) accessed, but too inaccessible for storing data that needs to be used. Optical disks offer an alternative that allows fast and easy access to any stored piece of information.

Optical disk drives also have a number of advantages over magnetic hard disk storage. Firstly, a major concern when using magnetic disk drives is the possibility of a head crash, resulting in lost data. Magnetic heads must be very close to the disk to work properly: they are typically only 8 to 10 μinches above the disk surface. When shock or vibration occurs, the possibility of a head crash is very real. Optical heads, however, may be as far as 2 millimeters away from the optical disk surface, making a head crash extremely unlikely.

Secondly, magnetic disk drives are much more sensitive to contaminants (dust particles, smoke, etc.) in the head assembly and on the disk itself. Particles on the disk or in the head can interfere with the reading and writing of data, causing the information to become corrupt. Optical heads do not focus on the disk surface, but on the substrate which is encased in glass or plastic. Because of this, contaminants have much less of an effect on the reading and writing of data to an optical disk.

Thirdly, optical disks are much less vulnerable to stray magnetic fields than are magnetic media. Once data is stored on an WORM optical disk, it cannot be erased by accidental exposure to magnetic fields. With erasable disks, the magnetic force required to corrupt the stored data at room temperature is much higher than any magnetic force that would occur in an ordinary environment.

Optical disks have additional advantages over all other storage techniques. A 5.25-inch optical disk can easily hold 200–400 MBytes of data (or more) per side. Optical disks can hold at least 400 Mbits of information per square inch, compared to 49 Mbits per square inch for magnetic media. The removable optical disk cartridges are compact, capable of storing vast amounts of data, and easy to access when you need information. And, unlike the higher capacity Winchester hard disks, the information is stored on a portable optical disk that is easily removed and transported from one machine to another. Optical disks are less expensive than other data storage methods when you take into account the overall data capacity, and with a WORM (Write Once, Read Many) drive, data cannot be accidentally (or intentionally) overwritten. No other data storage technique has this unique combination of features.

Disadvantages

Though optical disk drives have a number of advantages over other storage methods, they are not completely free from drawbacks. Speed is an important consideration in any application. Though optical disk drives are faster than sequentially accessed magnetic tape, their access times are at best comparable to magnetic hard disk. When speed is crucial, the optical disk drive may be outperformed by magnetic hard disk.

Presently, the most outstanding difference (which may or may not be regarded as a disadvantage) between optical disk drives and other storage techniques is that data written to a WORM optical disk is permanent. Once data is written to a WORM optical disk it cannot be overwritten, edited or erased. This is not the case for any other popular storage/archival techniques.

WORM Error Detection and Correction

The write-once aspect of WORM optical disks creates another problem: how to ensure data reliability in write and read operations. WORM disks, by their very nature, are more prone to surface defects than magnetic media. A typical optical disk has a bit error rate (BER) of 10^{-4} or 10^{-5}: meaning that for every 10,000 (10^4) or 100,000 (10^5) bits written, one error (on the average) occurs. A typical BER for magnetic hard disk is much closer to 10^{-12}.

On magnetic disk and tape, surface defects can be detected before the medium is ever used by writing data and then reading it back. This procedure identifies the defective sectors and tracks, which are then skipped over when the drive is used to actually store data. This greatly reduces the chance that data will be written on a defective area. This method cannot be used with WORM optical disks, since write operations destroy the disk. So, the quality of the optical disk surface cannot be ascertained before it is used. Certainly, high production standards can help to reduce surface defects, but such defects cannot be eliminated. Errror correction codes are the only way to increase the reliability of a WORM disk surface. Such techniques can reduce surface defect errors to 1 in 10^{12} bits written.

Errors can also be caused during the writing operation if the laser is poorly focused, or if the head assembly is not properly centered on the track. Optical disks are grooved, much like a phonograph record: writing is performed on the raised portion rather than in the groove itself. If the head assembly is not properly centered during the write operation, the pit representing the data will not be centered on the raised portion, and may not be read properly.

To compensate for the fact that optical disks have a relatively high BER, very effective error correction techniques must be employed. One method for detecting errors entails dividing data that is stored by a known polynomial, and storing the result on the disk. When the data is read back, the division is performed again, and the result compared to the result stored on disk. If the results differ, an error has occurred. While this method will reveal that an error has occurred, it cannot pinpoint the exact location of the error, and it does nothing to correct the error.

Another alternative is to use an error check and correction (ECC) code. With this method, errors are checked and corrected as they occur. This method is more useful since it actually corrects errors as they occur, but this method also uses a significant amount of disk space. This space is transparent to the user.

The biggest problem with any error correction method is that bad sectors of the disk cannot be forseen and compensated for. If a large area of the disk is defective, a lot of time and space can be wasted detecting and correcting error after error as it occurs. Despite these problems, errors can be found and corrected, without significantly deteriorating the performance of the WORM optical disk drive.

Applications

Optical disk drives, because of their ability to store very large quantities of data in a limited space, are especially suitable for data archival and backup puposes. In general, applications involving information that does not change over time can take advantage of WORM optical disk technology. This includes such things as testing results, seizmographical data, topological data, medical data, and knowledge bases. With the advent of erasable disks, the applications will grow to encompass any area where very large amounts of information must be stored as compactly as possible.

Meeting Military Specifications

Many military applications, such as terrain guidance systems, moving map systems, and data archival and backup, make use of unchanging data. With such a range of uses for optical disk technology, meeting military standards (not an easy task by any stretch of the imagination) is the next obstacle that must be overcome.

Different branches of the military have different specifications for the equipment they use. Standards must be met for every aspect of a drive's function, including radiation, noise, shock, vibration, temperature, humidity, and voltage. A drive that will be used on a submarine may have to fit through a hatch that is 25 inches in diameter, and be able to endure pitch and roll 30 degrees from horizontal with no loss of function. One shock test for the Navy involves dropping a hammer onto the drive during operation.

Optical disk media may well have the most trouble standing up to military guidelines. Two different substrates are considered for use: glass and plastic. Glass is usually considered a better choice as it can tolerate higher temperatures and more vibration. Most optical disks employ a sensitive material that is Tellerium based. Tellerium, however, does not tolerate humidity well, making it susceptible to corrosion. To compensate for this the medium must be sealed within an air pocket, which can cause problems if the air pressure drops. Some companies have turned to a Platinum-based disk which is not as easily corroded.

Software Support for WORMs

WORM optical disk drives do present some interesting challenges in the area of software support. As an example, consider Hewlett-Packard machines and operating systems. All HP operating systems currently in use require a storage medium that allows multiple writes to the same track or sector. Directory entries, which reside in clusters on the disk, must be updated each time a file is added or modified. WORM disks cannot be written to more than once, so the directory cannot be maintained using conventional methods.

There are two basic solutions to the WORM access problem:

1. Access can be limited to disk imaging, which allows a volume to be stored for reading access only;
2. Special drivers and software can be developed which allow a complete range of operations to the optical disk.

Disk Imaging

Creating a disk image for read-only access requires no special software tools. With this method, the user can copy an entire disk or volume onto the optical disk, using the appropriate commands from the operating system being used to access the disk. This method creates an image of the disk being archived. Once the disk or volume has been copied onto the optical disk, the copy can be accessed using the same commands that were used to access the original disk. This method can be used to make an exact duplicate of an existing volume with any type of directory system.

The drawback to this method is the fact that access to the copy is "read-only" access: new information cannot be written to the optical disk once a disk image has been made. Also, making a disk image will use up an entire optical disk volume. If the optical disk drive being used to hold the copy cannot be partitioned into smaller volumes (either through hardware or software), a lot of space may be wasted. Making a copy of a 40 MByte hard disk on a 200 MByte optical disk will use up the entire 200 MBytes, unless the disk can be partitioned into smaller volumes.

Read/Write Access

The software problems for supporting WORM optical disks are centered around the fact that a given sector of an optical disk can only be written to once. Conventional operating system support for directories requires the ability to rewrite a given sector within the directory area. To circumvent this restriction, a new directory structure must be devised which eliminates the need to rewrite sectors to support an elaborate file system.

OPTIDAM (for OPTIcal Directory Access Method), developed by IEM for use with its optical disk drives, is an example of such a directory system (see Figure 3).

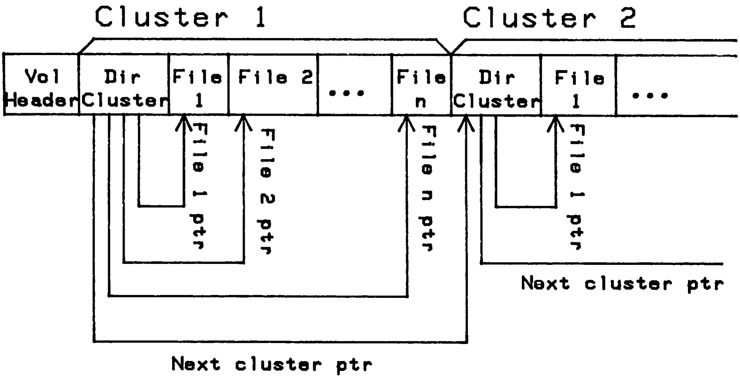


Figure 3: OPTIDAM Directory Structure

Each time the user writes a file or group of files (a group of files is a set of files open at the same time), information about these files is placed in a cluster of directory entries, and then written into a single sector which is located physically before the files on the optical disk. This is shown in Figure 3.

Each cluster represents a group of files that were opened at the same time by the user. The directory cluster is not written out to the disk until the last file that was open for writing is closed. This way, more than one directory entry can be written on one sector: the directory sector is not written until all the files are closed. It is conceivable to have only one file per cluster, but only if the user opens one file for writing and closes it before opening another file for writing.

Before each cluster is written out to the disk, sufficient information is written into the cluster structure to allow for finding the next cluster on the disk. This is accomplished in part by a pointer which points beyond the last file in the cluster. Each cluster contains one directory entry for each file in the cluster. This entry has information similar to any entry used by other operating system file supports such as LIF or UNIX files.

When the OPTIDAM directory structure is being used, there are two ways to support access to the optical disk: using existing operating system commands, or using archival utilities.

It may be possible to use the existing operating system commands to access the disk, with a few restrictions. Obviously, commands that violate the WORM nature of the disk (such as the BASIC commands PURGE or RENAME) cannot be used. This solution has the advantage of familiarity to the user, who is not required to learn a new language to use the disk drive. Alternately, archival utilities can be used to access the disk for reading and writing. Such utilities need to take into consideration the WORM nature of the drive, and the workings of the operating system.

In summary, access to a WORM optical disk can be supported in three basic ways:

1. **Disk Imaging.** This requires no special software tools, and allows users to make backup/archival copies of an existing volume (with any type of directory system) for reading access ONLY.
2. **Archival Utilities.** With archival utitlies, "new" commands are defined with special software, and are used to access an optical disk for reading and writing.
3. **New File Systems.** As an alternative to archival utilities, software can be written which employs a new filing system. Ideally, the new file system would allow users to access the optical disk using existing (and thus, familiar) operating system and language commands.

The Optical Future

The technology of optical disk storage is a very new field, and is expanding rapidly. As research in this field continues, there is bound to be a barrage of new developments to enhance and improve the science. Once erasable disks become widely available, users can look forward to the development of "direct overwrite" optical disks. With direct overwrite, users will be able to overwrite information on an optical disk, rather than having to erase and rewrite, as with current erasable disks. As the science grows, disk capacities will grow tremendously, data transfer rates will increase, and the drives themselves will become more compact as miniaturization reduces the size of the disk heads and other parts. With such an exciting and rapidly growing technology, the future is sure to hold much in store for those who take advantage of the optical opportunity.

Automatic Identification and Bar Coding: Promise and Pitfall

by
Kenneth L. Kimbrough
Quality Consultants, Inc.
1775 The Exchange, Suite 380
Atlanta, GA 30339

and

David Wiedman
Accuscan, Inc.
2179 Northlake Parkway, Suite 108
Tucker, GA 30084

Introduction

The implementation of automatic identification systems promises to bring significant improvement in control and productivity to American Industry. However, as with any new approach that promises to improve things, at the outset you can only be sure of one thing; change. Fortunately, data processing professionals are already experienced with how to implement systems. In other words, may of the lessons of the past can be applied to the future. One of the first lessons of any data processing implementation is learn as much as possible about the technology at the outset.

That is the purpose of this presentation; to pass on to you some of the knowledge you will need in order to successfully implement automatic identification solutions.

Any project that includes barcodes should address these five fundamental areas.

* The barcode symbology

* Printing of the barcode

* Verifying the Quality of the Barcode

* Reading the Barcode

* Collecting the Data

To put barcoding and automatic identification into perspective, it is important to know a little of its history and to briefly examine some more notable implementations.

The Supermarket

I believe it is safe to say that barcoding was born and raised in the supermarkets of America. With the advent of the concept of the self-service grocery store in 1916, the ground was laid for barcodes. When self-service grocery stores came along, the requirement to individually price items was also introduced. Before self-service, the price of items was controlled by the merchant by denying customers access to goods for the most part. Prices were not always on items and were generally stored in the owners or clerks head. The whole point of allowing customers to serve themselves in grocery stores was to reduce the cost of distribution. Indeed, from 1934 to 1974 the cost of distribution as a percent of gross sales was reduced from 24% to 20.9%.

However, as the number of items being sold, the numbers of customers being served, and the cost of labor increased, the cost of distribution began to go up again. Fortunately, the technology to address the problem was being developed.

In 1967 the first pilot barcode check out system was installed in a Kroeger market in Cincinatti, Ohio. The pilot was generally a success but one of the findings was that for the system to work, a uniform standard of barcoding products was necessary. Thus the Universal Product Code, or UPC was born. We will look more closely at UPC later in the paper.

Automatic Document Tracking - Federal Express

Federal Express is a company that has captured the lions share of the express mail business in part because of the integration of barcoding into their business. Federal Express gambled that they could gain the competitive edge in their industry through not merely adding barcoding onto their existing system, but but making it central to their business. Today, Federal Express uses the barcode to track every package they handle. And they handle a significant amount of packages. The most currant figures indicate that Federal Express processes 800,000 packages every night.

The Federal Express systems works as follows:

Every package that is mailed is accompanied by an airbill which has an 11 digit barcode symbol. When the courier picks up the airbill, the barcode is scanned using a hand held portable scanner and certain information is keyed into a portable data collection device. By the way, Fed Ex realizes an accuracy rate of 99.9999997%. Most errors are traceable to data entry errors and not scanning errors.

Once all the items being picked up at a particular site have been entered, the hand held scanner is inserted into a Radio Frequency unit in the pickup van and all the pertinent information regarding the airbill is transmitted via relay sites to the central data processing center in Memphis, Tennessee.

Each shipment of barcoded airbills is statistically sampled for scannability. Today Federal Express says that an airbill is scanned an average of 16 to 17 times from the time they pick it up until it is delivered.

LASER SCANNING OF MEDICAL SUPPLIES

The hospital of today is very concerned with accurate information with regard to the processing of laboratory work and dispensing of medications. The term that is used is "zero error". In many situations there is no margin for error!!.

More and more hospitals are turning to barcodes to provide the accuracy and timeliness necessary to meet their stringent requirements. One implementation that has emerged involves placing a barcode on each patients identification band. Each time a different service or function is provided to the patient, the barcode is read and associated with the transaction. Data about the transaction is then fed in either batch or real-time mode to a patient accounting system.

In addition, barcoding applications in medicine are found in tracking of lab work, blood (both samples and plasma for transfusion), and critical records such as X-Rays.

Inventory Control

Inventory control is another classic barcode application. The location of an item can be barcoded, as well as the item itself. Boxes of items can be barcoded, and the information can be captured with portables using contact or laser readers and transmitted in batches or even interactively through a short range radio transceiver. The application of radio transceivers is usually referred to as RF or Radio Frequency.

The end result is that the company knows what types of products they have on hand, how many and where they are located.

Barcoded Personal Identification

The barcoding of personal identification badges is an area of growing importance and utility. The most commonly encountered applications involve the addition of barcoded student or employee numbers to a permanent badge In the case of student ID's, the barcode may be read at the cafeteria cash register to charge meals to eating plans or checking out a book from the library.

Barcoded employee badges have a variety of applications ranging from tracking time and attendance or tying a manufacturing process to a particular employee as well as the more common uses of security.
For example, a barcoded "traveller" can be created which identifies each station during a manufacturing process. As the traveller reaches a station, it and the employees barcode are read. When a step is completed, both barcodes are read again and the Shop Floor Control system is updated. The data collected can be used to arrive at more accurate forecasts of production time requirements and to pinpoint areas for improvement.

Automobile Identification

The next time you use a rental car, notice that it probably contains a barcode. The barcode is being used to keep track of such diverse activities as rental check in/out, preparation for rental, and periodic maintenance.

In addition, the three major automobile manufacturers are implementing a system to put a barcode not only on every part of the car but on the car itself and track the car with that unique serial number that is represented in a barcode format (some of the rental cars do that too).

Fixed Assets

Another major area for barcoding where significant labor savings can be realized is with conducting fixed assets inventories. In a typical barcoded implementation, each fixed asset is tagged with a barcoded/human-readable label. At physical inventory time, each item is read with a portable data collection device similar to the unit in use by Federal Express. As each organizational units inventory is completed, the data is passed to a database where a comparison is done and reconciliation reports are created.

In an organization where large numbers of items must be accounted for or deprecitation rules dictate accurate inventories, these systems will more than pay for themselves with the first inventory.

A central theme runs throughout all the applications of barcoding technology that we have seen so far. These systems deliver

productivity,
control,
competitive edge.

All of these systems provided some form of improved productivity. Manually keying in data takes time and has a much higher degree of error rates.

People can accomplish more in situations where barcoding has been properly applied. Inventories and items can be tracked faster and more accurately. This all adds up to better control.

The first year of operation of Federal Express's Customer Oriented Service and Management Operating System, $10 million was saved in billed revenue that had previously gone unbilled.

Federal Express can make the claim that they can deliver documents overnight anywhere in the U.S. and at any time tell you within an hour where your document is. That is a competive edge!!

THE BARCODE SYMBOLOGY

One of the first considerations when implementing a barcode system is the choice of symbology. This step is critical because the symbology selected will be used to represent your data throughout the system.

There are over 50 major symbologies available or in use today. Among the most commonly used barcode symbols are:

UPC
Code 3 of 9 (aka Code 39)
EAN
UPC
Codabar
Interleaved 2 of 5
Code 11
Code 128

Barcode Specifications

A barcode consists of a number of printed bars and intervening spaces. the widths of the bars and spaces, as well as the number of each, is determined by a specific convention, referred to as a specification for that symbology. A specification sets the following conditions:

* the minimum nominal width of the narrowest elements

* the ratio of the wide elements to the narrow ones

* the printing tolerances (the changes in widths of bars due to the printing process

* the structure of unique bar and space combinations to represent various characters

* the bar/space patterns that signify the beginning and end of the bar code message

* and the clear area or quiet zone, required in front and at the end of the symbol.

If these conditions are met the barcode is said to be "within spec".

The UPC Symbol

As the grandaddy and most widely known, the UPC symbol makes a good illustration of what makes up a barcode symbol. The UPC symbol is made up of a ten digit number. Twelve digits if you consider the first character and the last character. Ten digits because ten digits are represented along the bottom. What that UPC code represents is a manufacturer ID and a product ID. Price is not represented in that UPC barcode. The first character that you see which is about half-way up the barcode on the left is whats known as a system character. This represents what type of item this is in a retail environment. If it is a zero (0), it is a retail item. If it is a three (3), it is a drug item. So if you were to go into an Eckerd's or Treasury Drug and buy a box of aspirin, you would see a three (3) there. Yet if you were going to a supermarket and pick up a can of coke you would see a leading zero (0). Now, this is a UPC Version A which is ten digits. UPC works this way. Each letter is represented with two bars and two spaces. Those bars and spaces can be one of four different widths. Those series of bars make up the number then. The first five digits you see on the bottom represent the manufacturer. Lets take the Campbell soup example, the first five digits represents Campbell's soup, the second five digits might represent a can of tomato soup. No price is represented in that barcode when that scan at the point of sale that accesses the price. UPC is just one type of symbology that we talked about.

The "X" Dimension and Code 3 of 9

Code 39 allows you to represent alpha-numeric data in the barcode, whereas UPC is very fixed strictly numeric and strictly for the retail type environment, Code 39 is much more flexible.

The X dimension is the narrowest bar element. In any typical barcode there are two element sizes. In fact the name Code 39 is derived from the fact that any character is represented with nine elements and three of those nine elements are always Y while the remaining elements are the narrow or X element.

Having established what the X dimension, or our narrow bar element, is, we then establish a ratio called the Wide to Narrow Bar ratio. With Code 39 the Wide to Narrow Bar ration can be anywhere between 2.0 to 1, to 3.0 to 1. Or, put another way, if you have a narrow bar element that is 10 mil across and a 3 to 1 ratio, then the Wide bar would be 30 mil. Those would be your two element sizes. You can see here in these barcodes I have represented the same word from top to bottom, yet The bottom barcode is very very long. The top barcode is very short, very dense, yet they contain the same data. The reason why is because I've varied my narrow bar size, I've increased it and I've also increased my Wide to Narrow Ratio. As I affect both of those variables, I affect the density of the barcode.

PRINTING THE BARCODE

Once the symbology has been chosen, the next issue concerns printing. Printing is really the foundation of our barcode system. The printed symbol is crucial to everything that lies beyond it and that encompasses scanning, and data collection. Because so much depends on it, the symbol must be printed well. Unlike human readable text, there is much less margin for error as we shall see later. It is very critical that bars and spaces are the correct width. If those widths get out of spec in any way it will affect the read rate and accuracy of our scanner. Because the quality of the barcode is so important to a barcode system, a company should ask itself is "Are we going to take on the responsibility of printing the barcode or are we going to have another organization print for us"?

However, there are some other considerations when determining whether to print your own barcodes or not. If the information to be printed is known well in advance of the need to apply the information and the quantities to be ordered can obtain volume discounts, then outside printing is recommended. For example, in the case of Fixed Assets at HP; the asset number is a sequential number assigned to the asset as it is received. High quality, laminated, barcode labels can be ordered and distributed from a division or region office. The cost of distributing the means to print barcodes at remote locations, in the case of a sales region, is not justified. In addition, the sequential number can be tied to the items serial number and value through the Fixed Assets Database. The situation with Fixed Assets is very similar to the Grocery application. There is no need to code value (or price) on the unit itself.

However, in situation where the information to be coded is not easily known beforehand, on-site printing may be recommended. Let's take a typical manufacturing environment for example. When a finished good is produced, it may be assigned a serial number.

Typically, one of the most popular off-site methods is the offset printing. With this method, a master of the barcode (and perhaps other information) is created and a film master is created. A major advantage of this method is that the barcode or label being printed can be of the highest possible quality.
In addition, there is no need to purchase hardware or to train your staff to run a barcode printer.

Some limitations are (1) that you have to depend on someone's lead time to produce the labels and deliver them to you, (2) you may not know what information needs to be printed until the labels are needed, and (3) it may be more cost effective to control printing in-house.

An additional factor when confronting the issue of in-house versus outside printing is the availablilty of high quality, reliable printers today.

On todays marketplace there are essentially three methods for printing barcodes, dot matrix, laser, and thermal.

One of the most widely used methods of in-house barcode printing is with the dots matrix printer. With dot matrix there is a hammerbank in the printer and which contains hundreds of small hammers. The hammers are a certain dot sizes which determines the smallest dimension barcode that can be printed. When the hammers strike the paper they form the barcode from top to bottom.

Some advantages of dot matrix printers are that they are cost efficient, fast, and they can handle a wide variety of paper and label stock sizes and thicknesses. One disadvantage concerns the quality of the barcode printed by dot matrix.

Because there is a physical slap of a hammer on the paper, the dot itself can be distorted. In addition, since the hammer is round, a close examination of even a high quality dot matrix barcode will reveal a wavy pattern.
If the printer is not regularly serviced, chances are that the barcode will eventually go out of spec. Once the barcode is out of spec, the bar code read failure rate is going to increase. In other words, the barcodes being printed will be come useless. The purpose here is not to discourage you from considering dot matrix printers. Rather, you simply must be aware that you are using a mechanical device that will require more frequent periodic maintenance that you might expect from a the ordinary system printer.

The Thermal Process

Another commonly encountered printing process is the thermal printing. There are two types of thermal printing; thermal and thermal transfer. Both processes produce a very high quality image, very fast and are cost effective. Both thermal and thermal transfer involve the heating of a print head. They differ in that thermal printing the heated print head is pressed against chemically reactive stock. Whereas, with thermal transfer the heated print head is pressed against a film stock. The portion of the film stock representing the characters is transferred to ordinary paper/label stock.
With either thermal process a very high quality image is produced.

The major drawback to thermal printing is environmental. Because thermal printing uses a heat sensitive paper stock, it can be easily discolored if stored or used in hot environments. If thermal stock is stored in a very hot truck, there is the potential for the entire label to discolor. Retailers will use this type label very often because they work in a controlled environment.

Two other issues need to be considered when using either thermal process; speed and label size. Typically, thermal printers will not handle stock over 8 inches wide. Printing speeds for thermal printers are usually in the range of 2 - 3 inches per second.

Laser Printing

The past few years have seen the emergence of relatively low cost laser printers. Because of some inherent technical difficulties, however, laser printers have not yet had a significant impact on the printing of barcodes. Given the growing demand for barcodes and the advances in technology this picture will likely change in the next decade.

To understand why laser printed barcodes have not been more pervasive, a brief explanation of how laser printers work is in order. Laser printing uses the same basic technology found in photo copying machines. Thus the correct name for laser printing is Electrophotographical Printing, or EPG. In a typical EPG printer, employs a Helium-Neon laser to "draw" the characters on the paper/label stock. The laser causes an electrostatic image consisting of charged and uncharged areas to be formed on the surface of an cylindrical drum. A toner consisting of small, black, electrically charged particles is brought near the surface of the drum, and the toner particles adhere to the paper.
The importance of understanding this process is that two conditions exist which can affect your application. First, high speed laser printers (40 pages per minute and up) use several drums and can run quite hot. Special consideration must be made when evaluating the gummed-label stock to be run through the printer. As you can imagine using the incorrect stock can result in frequent jams and unreadable output. Second, because laser printing causes toner to be electrostatically adhered to the print stock, there is no absorption of the printed image into the paper. If the barcode is to be read using a contact reader, special care must be taken to ensure that the barcodes don't smear. Paper stock is available that will prevent smearing but you must look for it.

SYMBOL VERIFICATION

Having established the method for printing an often overlooked aspect of barcoding must be addressed; symbol verification. The entire purpose of verification is to ensure that the barcodes you are printing can be read. If the symbols are unreadable, you have defeated the purpose of the system. Simply reading the barcode with a standard scanner is usually insufficient; all you have proven is that your reader will read that particular symbol. To ensure that the printed symbols are in spec they must be read by a verification device.

A verification device evaluates the "X" and "Y" dimensions of each symbol. Some devices also calculate the wide to narrow ratio and checks that the correct print contrast is maintained.

Print verification readers range in cost and sophistication from the type used by an offsite printer in the $10,000 range to devices in the
$1,500 or $2,000 range that merely check the barcode and tell you if it is in spec.

SCANNING

Now that you have printed a readable, verified barcode symbol, it has to be scanned to be of use. Before looking at scanning devices, a few words are in order about the physical act of scanning.

With any type of scanner that you use, the symbol must be scanned completely. The scanner must move all the way through the barcode symbol. You can't stop in the middle, lift up and go back down and go from there. Also, it doesn't matter if the symbol is read from left to right or right to left. Barcodes are always omni directional. Every properly printed barcode has a start and stop character in the barcode which identifies which direction it is being scanning from. The important thing is that a full pass must be made through the barcode symbol.

What the barcode scanner is looking for when it passes over the barcode symbol, is a pattern. What the dot of light is trying to encounter is a pattern of dark and light bars and spaces. When the light encounters a space, more light is reflected back. When the light encounters a bar, there is very little reflectance.

Types of Scanners

There are only two types of scanners available on the market; contact and non-contact. The best known example of a non-contact is the slot scanner found in supermarkets and some retail stores. With non-contact scanners the operator does not have to make contact with the barcode. To achieve this objective, the supermarket slot scanner is actually over a hundred different light beams. In order for the barcode to be read, at least one of the beams must make a complete pass over the barcode. Because of the sophistication required to make this type of reader function properly, it is quite expensive. An alternative device that is usually more affordable, is the light pen.

Light Pens

Initially, nearly all barcode reading was done with a contact reader known as a light pen. Light pens are alot less expensive, more in the $150.00 to $200.00 price range.) What they allow the operator to do, is make a pass across a barcode and look at that symbol one time. A consideration when evaluating contact versus non-contact is the reading surface. If the label is affixed to a variable surface, it is going to be much more difficult to read than if the label is affixed to a smooth, flat surface.

Laser Scanners

In contrast, a non-contact scanner may cost 6 to 7 times as much as a light pen but it has certain advantages over the light pen. A laser scanner works by producing a dot of light using a mirror mechanism that rotates inside. Like the light pen, it also produces a dot of light. However, the dot of light is moving back and forth at forty scans per second. To the human eye the beam looks like a line. In reality, the line is the same dot being bounced off a mirror very swiftly. Because the dot is being scanned forty times a second, there is a much higher probability that the barcode symbol will be read by the scanner that by a wand. Therefore, you will see or hear the term "aggressive" used to describe a laser scanner. Two other advantages to laser scanning are that items can be read from several inches to as much as 7 feet away and laser scanners have a much higher success rate with reading irregular surfaces.

Fixed Mount Scanning

Occasionally, the situation arises where fixed mount scanners are called for. These are always non-contact readers. They are best utilized where it is either very easy for the operator to manipulate the object to be scanned or if the movement of the scanned object can be tightly controlled. Two common examples involve fixed scanners for reading symbols in library check out and scanners reading labels off of boxes on an assembly line.

DATA COLLECTION

Two major methods of data collection are employed today. The first is fixed terminal/station. The second involves the use of portable data collection devices. In both cases, contact and non-contact readers can be used.

Fixed Station

Typically, a fixed station has the following components:

- a barcode reader
- a digitizer (aka "wedge")
- a personal computer/work station or terminal

We have already discussed the barcode reader and most are familiar with the personal computer/workstation or terminal configurations. Which leaves the "wedge". The purpose of the wedge is actually quite simple. The data reader by the data collection device is in analog form and must be translated into digital form before it can be used by a digital computer. The wedge is the translator. As technology progresses, the wedge will probably disappear. For example, within the last year Hewlett Packard has introduced a wand with the wedge built into it. The need for digitizing hasn't gone away. On the contrary, only HP has simple miniaturized the wedge and put it where it belonged, into the wand.

Portable Scanning

Portable scanning encompasses the use of any device not physically attached to the "data analysis" computer. The most commonly encountered methods are portable data collection terminals, such as the hand held unit used by Federal Express, and RF transmitters. RF systems differ from all other applications in that once data is collected by either laser or light pen, it is transmitted to the host computer via Radio Frequency. Thus, the individual doing the dat collection can be a mile or more from the "host" and each barcode that is read is immediately recorded without the need to upload at a later time.

Another form of RF data collection involves the use of "RF ID Tags" attached to items. In a recent application, RF tags are attached to the chassis of automobiles at the beginning of a production line. As the chassis travels through the manufacturing process, remote sensors read the information on the tag and keep the Shop Floor Control data base updated with the current location of the chassis.

CONCLUSION

At this point, most of the major issues involved in the implementation of a barcode system have been addressed. The last issue concerns the integration into your data processing system of the data you are collecting.

The most basic principal of data processing especially to barcoding. Garbage In Garbage Out!! Make sure that you are collecting the right information and that the data is good. Verify the barcodes!! How will I or my users use the information we are now collecting. Remember, you may now be collecting data in quantities you never dreamed of before. Be prepared!!

I began by saying that the lessons you've learned in years of implementing data processing systems can be applied to barcodes and that the first thing you had to do was to learn about the technology. Hopefully, this overview of barcoding technologies has given you some food for thought. For those of you just starting out, you have just completed your first lesson. There will be no test today!

SYSTEM SECURITY?
As Soon As I Can Find The Time...

Steven G. Bloom
InCase Corporation
2055 Woodside Road, Suite 171
Redwood City, CA 94061

INTRODUCTION

Why is security so often talked about but just as often found at the bottom of the list of priorities? What we will be discussing will not be state of the art, brand new, never known before stuff about locking up the security on your HP 3000. You have probably read the many articles that have been written or will have heard the many talks given on how to close those nasty security holes. We will be reviewing some of these gems in this talk but my hope is that perhaps in understanding why security is so low on most of our lists, and realizing that it should be paid more than lip service, some system managers or MIS managers will return to their sites and take whatever action is necessary not next year, not even next month. Start reviewing your security procedures immediately! DON'T wait until your company becomes one of the fast increasing computer crime statistics! Although not nearly as fun as performance tuning, and certainly more likely to ruffle than smooth feathers, taking action NOW may save your company from a tremendous loss.

You might be surprised to discover how many system managers believe that they are safe from unauthorized intrusion even though they have taken no measures to protect their systems from unwanted access. Why do they believe that they are safe? Some state that their machines are only used by a very small group of people and that these people can be trusted with the data found on their HP 3000. Others say that this is merely a development machine and therefore, there is nothing to protect. Hmmm...

In this paper, we will be looking at the issue of security from the following perspectives:

- The history and growth of the HP 3000 business environment as it relates to security
- Specifics - What should be examined in the typical HP 3000 site?
- What is a security review? Who should perform it?

What are the measures being taken in HP 3000 shops today? In fact, they range from those mentioned previously, where even passwords are considered to be too much bother to the users, to shops in which only one user has access to an SM USER ID and every other user (with the exception of the single operator), logs into a tightly controlled, menu driven system with no access to the MPE colon (":") prompt. It is my perception that security concerns are of a lower priority in many HP 3000 shops than in facilities using other machines, such as DEC and IBM.

HISTORY

How did we get to our current state? To answer that question, we can look at the life cycle of the HP 3000 as a product line and its life cycle within a typical business environment.

Even in its early days, the HP 3000 was brought in to solve problems that were smaller in scale than those requiring the bigger "mainframe" computers. Then, as now, it was perceived as a "friendly" machine, not requiring an army of data processing professionals. An MIS department might only require an operator and perhaps a programmer. Due to its relatively low overhead requirements, the HP 3000 was (and is) seen as an excellent choice for a small company or division. These small companies often felt like "family" since everyone knew everyone else and everyone wore many different "hats". This contrasted to the big IBM shops where the complexity of the machine and of the operating systems required on-site system programmers and specialists in all aspects of system usage, including security.

ALL IN THE FAMILY

Many of those small companies that used the HP 3000 did not stay small. As the HP 3000 matured and grew more powerful, an increasing number of applications became available to the users. The users who once numbered in the tens were now pushing against the upper limit of an operating system that could handle 80 plus users at a time. Technical support staffs also grew and gradually, everyone no longer knew everyone else. The family had become a "community". However, as these small companies grew into bigger companies, no one wished to give up that feeling of "family", and so, there has been tremendous resistance to implementing policies and procedures that might have been considered as divisive or intrusive, such as security.

The resistance to implementing solid security was not only a result of the users' desire to maintain that warm fuzzy feeling of being part of a "family". In sites where "user ignorance" was the key to data security, even those technical support staff who agreed that the time had come for increased security measures responded with disdain at the notion that they should be locked out from total system access. I have heard this from MIS managers who tell me that they are not concerned about improving security because the only ones with access to the MPE "colon" prompt are the programmers, and they, of course, must have access to all capabilities on the system. I wonder how many times a bug fix was overwritten by an enhancement or vice versa. I wonder too, how often this might have been prevented if carefully thought out change management policies and procedures had been implemented which could be enforced only by maintaining strict security on the system, including restricting programmers' access of a system.

The power of the HP 3000 has grown as well, and with that growth, there has been a parallel growth in the size of the installed base. Many of the early programmers and operators gained increasing knowledge of the operating system and its strengths and weaknesses. Some went on to create applications and utilities that would make the HP 3000 an even more attractive machine to solve a company's data processing needs. Some of these applications were sold and others were shared with other interested HP professionals at user group meetings. INTEREX formalized the software library and conference swap tapes. And of course conferences have permitted users to share knowledge through its speakers and through other contacts. Articles in trade magazines such as INTERACT have also played a major role in sharing the wealth of information and knowledge gained about MPE.

KNOWLEDGE AS A DOUBLE-EDGED SWORD

Unfortunately, the dissemination of knowledge that has occurred as a result of sharing information at user group meetings and through the contributed library has been a double-edged sword for security. Why? Because the CSL tapes and swap tapes are FILLED with privileged mode programs. Each one of these programs is a potential "trojan horse". Conference speakers on security love to talk about security holes in MPE and WELL THEY SHOULD! But again, knowledge can work for you or against you. While providing system managers with the information they could use to ensure a relatively secure system, the articles and talks also provide other technical people new ideas for hacking into a system that they can try out as soon as they get the chance.

The point here is that two parallel phenomena have occurred during the growth of the HP 3000 installed base. The first is that the interactive "friendliness" of MPE has helped to propagate the lack of concern for security in various ways. Although HP 3000 sites no longer are perceived as only being "small shops", we still like the warm fuzzy feeling that we are all "family" and so there is no reason to lock the door. Secondly, technical knowledge and understanding of MPE has exploded over the last fifteen years within the HP 3000 community as a whole, and has provided many the technical ability to break into minimally protected systems.

I JUST CALLED TO SAY "I LOVE YOU"

In addition to these two factors, another security-smashing monster looms high over our vulnerable data. DIAL-IN PORTS!! Communications from the field is no longer just "nice-to-have". For many companies, it is imperative that the field offices, sales people, service reps, etc. are able to access and update the necessary data in the system. Use of dial back modems is certainly to be encouraged but is not practical in many instances. How, for example, is the field representative going to be "dialed back" at a pay phone on Route 66 or at the customer's computer room? This means that unless an operator is standing by 24 hours a day, ready to UP or DOWN a dial-in port on request from the field, the system is vulnerable to anyone with a PC and a telephone. For all intents and purposes the system manager should treat the system as if it were attached to terminals at the local community college, which is virtually the case!

WHAT IS THE SOLUTION?

You aren't going to like the answer. There is almost nothing that can be done that will provide both an absolutely secure system, while at the same time allowing everyone, from programmers to users, the flexibility they all want. I believe that the answer is in INFORMATION. The system must be reviewed as a whole. Its strengths and weaknesses in terms of security must be weighed against ease of use and functionality as well as cost of change. You or your system manager must do a full security audit or review of not only the operating system security but of procedural and physical security as well.

Lets briefly review the three major areas of concern regarding security, physical, procedural, and logical.

Physical (or "Things that someone can do by wandering around")

- Is your CPU under lock and key?

  Access to the CPU or console is equivalent to access to your data for a knowledgeable intruder. If possible, the CPU, console, tape drive, and system printer should be in a room whose access is restricted to the minimum number of personnel.

- Is physical access to terminals restricted at all?

  Are any terminals in an area accessible to the general public? If there are, can this be changed? Can the ports to these terminals be DOWNed except when required?

- Are your modems accessible without either Dial-back or Operator control?

  The question to examine here is whether the outside world has access to your data? Are your modems physically accessible? Can an intruder simply "tap your line" at the modem itself? And most important, can a high school kid with a PC and a telephone get access to your data?

- Is access to the system printer restricted?

  Access to the system printer should be restricted, if possible. Valuable information should not be available to anyone with sticky fingers. Printouts should be kept in "lockboxes", available only to the user to whom they belong.

- How well is your tape and disc media protected?

  Tapes and disc packs kept onsite should be kept in a locked, fireproof room. Access to the key to the tape store room should be restricted to trusted individuals who have a bona fide need for access. Backup tapes should be stored offsite.

  Tape management is not just a disaster recovery issue. Accounting information is kept on a SYSDUMP tape which, if perused by someone who knows the tape layout, can provide useful information to an intruder.

- Is the console kept under absolute control?

  Don't distribute console capability lightly through the :ALLOW command. Again, be sure that there is a bona fide reason for granting special capability.

- Is a disaster recovery plan in effect?

  Although not a security issue, disaster recovery should be a major concern in any MIS organization that is charged with the management of critical corporate data. What would happen to your company if its general ledger, accounts receivable, engineering, inventory, sales, order management and manufacturing information bit the dust? There are disaster recovery plans available in many forms. If you do not yet have one in place, create one immediately!

Procedural (or "Things that someone can trick others into doing")

- Does anyone besides the OPERATOR have the ability to perform a STORE or RESTORE?

  This is a very important procedural question. Although MPE provides some safeguards on control of access to the tape drive through the REPLY required when a STORE or RESTORE is requested, this does not completely protect the system from an unscrupulous malefactor. A STORE tape could be given to the operator with instructions to REPLY to the tape request from what the intruder claims is a RESTORE that s/he has issued. In fact, the intruder has issued an FCOPY from the TAPE device which, once the unknowing operator has REPLYed to the request, will transmit all kinds of accounting information to a disc file that the intruder can examine at leisure.

- Are passwords required for all users on the system?

  Although there are usually accounts that contain little or no protectable files, remember that any access to the system might be dangerous. Once someone is logged on, the possibilities are much greater that illicit access will be acheived. Password all users.

- Are passwords required to meet minimum standards?

  Minimum Length. Passwords should be of a minimum length. I believe that a five character password is the minimum that forces a nontrivial effort to obtain the password through trial and error.

  No Jumbles. Passwords should NOT contain any combination of the characters in the USER ID, ACCOUNT name or GROUP name that it is protecting. (ie: SELBAYAP to protect PAYABLES)

  Change Default Passwords. Default passwords which are provided for third-party software accounting structures MUST NEVER be used.

  Not Easily Associated. Passwords should not be easily associated with the user to whom a USER ID is assigned. Spouse's or children's names may not be advisable in a user community where everyone knows everyone's family intimately. Second cousin's names are probably fine.

  Not Too Difficult. Passwords should not be so difficult for a user to remember that the password will have to be posted on the terminal. When possible, permit the user to change their own password or provide it to the system manager if no password changing program is available or desired.

- Are passwords changed on a regular basis?

  How often passwords need to be changed will depend on the specific requirements of your site. However, at the bare minimum, they should be changed at least every three months.

- Are individual users assigned unique USER IDs?

  Accountablility of action is impossible when generic USER IDs are used. There are a number of issues involved here. Firstly, individuals who are given their own unique USER ID are more prone to be protective of their passwords. If the user is informed that the password is his/her responsibility, and that activity performed by someone logged on with that USER ID will be attributed to the user to whom it is assigned, it is much more likely that the user will make a greater effort to protect the password and not give it out.

  Secondly, as I noted just now, with a unique USER ID assigned to each user, accountability IS possible. Reading the system log files will inform the system manager what system activities may have occurred at a given time by which sessions or jobs. But if the logon for a session was USER.PAYROLL, the manager will not be able to trace activity to a human being.

Logical (or "Things that someone can do from a keyboard")

- Are you using the default logon error messages?

  MPE, in all of its infinite friendliness, teaches a non-HP hacker everything s/he needs to know about breaking into the system. Each step of the logon process is clearly marked with what kind of information is required of the individual logging on. For example, typing garbage at the initial colon prompt results in CIERR 1402, which informs the hacker that the word required at this point is "...HELLO, :JOB, :DATA, OR (CMD) AS LOGON". The system will continue to aid the intruder as the breakin continues with useful information such as the eight character maximum length of a name.

  This problem does not exist on a UNIX logon. No matter what is entered, the user is not informed whether s/he has gotten into the system until both a logon id and password has been entered. And even then, the user is not informed WHY the logon failed.

Unfortunately for the security conscious among us, this is not the way things were designed in MPE. But you can at least reduce the helpfulness of the error messages in the message catalog, by changing ALL of the messages to one uniformly unfriendly message such as "LOGON FAILED". (Terry Simpkins wrote an article on just this topic in the October 1987 issue of INTERACT. It is worth reading.)

- Do you limit SM, PM and OP capabilities to the absolute minimum?

Without going into too much of the gory detail, I will explain briefly the danger in each:

SM: A USER ID with access to this capability can access any file with virtually no restriction. This means that the user logging on with this USER ID can purge, modify, execute, lock and append to any file on the system. That user can also change accounting structures virtually at will and can gain access to privileged mode (PM) which provides even greater ability to destroy, maim and mutilate.

PM: A USER ID with privileged mode capability has direct access both programmatically and through DEBUG to all machine instructions. A knowledgeable user with PM is, in fact, more powerful than a user with SM, because the PM user can actually bypass the operating system, whereas the SM user is still bound by the laws governing the operating system.

OP: Although a less dangerous capability, the System Supervisor, (OP), capability still should be of concern to the system manager. A user with OP capability has the power to STORE across account boundaries. Remember that the STORE tape can contain some interesting information to the potential intruder.

- Do you limit PM capability for groups?

A group with PM capability is dangerous even if the user with access to that group doesn't have PM. A PM program can only be run while it is residing in a PM group. Therefore, any PM program residing in a PM group could harm your data or even the operating system if run. But the scary thing here is that the user running the PM program doesn't have to have PM in order for the program to execute its privileged instructions! The answer then is to make sure that file access to the PM programs in the PM group are limited or that the PM programs that are accessible to all comers are "safe", right? Wrong! The PM group must be writeable only by trusted users in its entirety! Read on...

- Is your system safe from "Trojan Horses"?

  A non-PM user can create a PM program! To do this, the user must simply compile and link a program containing whatever nastiness s/he likes. Then the user need only patch the file to set the capability flag for the program to PM. Voila! A new PM program! This also implies that ANY program file to which non-trusted users have both write and execute access is dangerous. If I can write to it with FCOPY, I can inject my own super program into a mild mannered program residing in any PM group, even one in another account.

THE SECURITY AUDIT

The security audit is the system manager's way of judging the efficacy of the security procedures that have been put into place. The security audit or security review is really a checklist that can be gone through at regular intervals. The areas to be reviewed are the items mentioned earlier in this talk as well as other items that are of particular importance in your site.

Determining a security policy and creating a security review checklist occur simultaneously. Take the time to develop this policy now.

Who should perform this audit? In some locations, the audit is done by the system manager, while in others an outside auditor is brought in to perform this auditing function. There are benefits to both approaches which we will briefly examine here.

The external security auditor has some distinct advantages in terms of the objectivity that can be brought to bear in an individual site. It is sometimes difficult to see the forest for the trees when you are trying to delve into your own system. In addition, scheduling an external audit forces the job to get done, whereas it is easy to procrastinate and put it off when you are the one that must do the job. As the subtitle says, "As Soon As I Can Find The Time...".

On the other hand, the internal auditor has the experience and knowledge of a particular site. The internal auditor can develop a site specific security review that deals with the reality of that company's needs and corporate style. Often an external auditor will provide an audit checklist that is so standardized that it has no relevance to the needs of the client.

If you determine that an external auditor is the way to go, try to find a company or consultant that specializes in the HP 3000. Although security concerns are generic to some degree, there are things to watch out for that are system specific.

OPERATIONS MANAGEMENT IN A MULTI-HP3000 ENVIRONMENT

Roberto Drassinower
Teresa Brzozowski
Carolian Systems International Inc.
3397 American Drive, #5
Mississauga, Ontario
L4V 1T8 CANADA

This paper discusses centralization and automation in a multi-HP3000 environment. It offers suggestions on how to plan for and introduce network-wide operations control, thus making large shops more manageable and efficient. This paper deals with centralization and automation as two distinct issues: where centralization is required before automation can begin.

Introduction: The Operations Optimization Project

In recent years, multi-system installations have become more common in the HP3000 community. While the advantages of both distributed and local network systems are often obvious, the challenges associated with their management as one coherent unit are all too often underestimated. Since each machine is typically monitored and controlled through its independent console, an overall perspective of network wide operations is difficult, if not impossible to obtain. Yet sensitivity to global issues and centralized decision-making are essential to the efficient management of large scale integrated systems. The problem is vastly complicated when a central site has to support remote sites. Often a lack of adequate operations staff limits control over production and severely impairs the flow of system status information back to head office. Similarly, the sheer size and complexity of local multi-machine installations can also render traditional operations management techniques ineffective.

The possibilities for streamlining networked operations do exist in an HP3000 environment. Careful planning can offer existing operations staff greater control over distributed systems from one location making them more efficient. At the same time, the resulting centralization of information provides management with a base upon which to build sensible network-wide policies and procedures for production. Such policies inevitably lead to improvements in overall system throughput and higher system utilization. Moreover, continually reviewing operations management systems guarantees the optimal use of existing resources as installations grow. What is needed, and is sometimes missing, is a firm commitment to an on-going, long-term operations optimization project. Although this will divert resources away from satisfying some demands of the data processing department, failure to review procedures may significantly erode its operating efficiency and ability to service the company in the long term. Since the results of your labour will be a long term solution, you can implement distinct phases without incurring prohibitive expenses, while at the same time remaining confident of a proper design and direction. In the long run, it will be well worth the effort.

In tackling the issues of centralizing and automating your DP shop, it is important to understand fully how your operations work and more importantly, how you would like your shop to work. Therefore, I would like to review some of the basic concepts about the operations environment before continuing.

Today, HP3000's are controlled by an operator through local console. This system console is the lifeline to the HP3000, since all operations status messages pass through it. It is a hub where all problems can be defined into two categories: those that require human intervention, and those that report an error condition to which someone should be alerted. In this scenario, it is assumed that trained operators are constantly monitoring the console screen so that they can respond to the requirements of the message immediately. This also assumes that the operators are trained in a variety of areas, so that they can handle job management issues, system security policies, internal system management, data comm, and so on. In short, good operators have to specialize in many different areas - a jack of all trades. Traditionally, the operators must also contend with a variety of messages which are irrelevant to their primary tasks. For example, if the operator is mainly concerned with executing and monitoring an installation's production schedule, then that person will only want to view job completion and error messages from among all activity passing through the console: logon messages, HPDesk messages, data comm messages and so on. The result is that the operator has to act as a human filter. And, as the site becomes more active or complex, the operator's job becomes increasingly difficult. The result is that often critical information will merely scroll off of a busy console and disappear, at which point the operator is faced with paging through hard-copy STDLISTs.

Now imagine this task in a multi-machine environment. Busy shops might need an operator per system in order to keep operations running smoothly. Keep in mind, however, that this setup makes centralization difficult by the fact that each console is physically separate from the next. Messages coming from multiple machines cannot be grouped logically. And an operator always has to be at the console to respond to messages. This can mean a lot of wasted time for a number of operators, since they are only concerned with a small percentage of the total messages.

The First Step to Improvement

Throughout this discussion I will be making general recommendations which, when grouped together, provide the basis for a long-term operations review plan. The first of these is as fundamental as it is self-evident: commit resources to examining existing operations practises through a long-term operations optimization project. It needs to be an on-going project so that your operations optimization can accommodate changes in demands.

As I alluded to earlier, the most substantial obstacle this kind of project must contend with is the difficult task of providing operations staff with both the information and control required to efficiently oversee multiple systems. Crucial to overcoming this problem is to determine the nature of these needs precisely, such that they can be satisfied as efficiently as possible.

As you may have guessed, the issues that must be confronted during such a study are not unlike those a systems analyst must wrestle with while designing an order entry system. For example, this individual would be concerned about the kind of system access data entry personnel require and the content of management reports. Quite clearly, different individuals often will require highly specific information which is of little consequence to others. On the other hand, recognizing common needs is just as important to the design of an integrated system.

In designing operations management systems much the same is true. Operators will need access to the consoles of all systems. In addition, particular operators may only be interested in a subset of all console messages. An operator charged with servicing tape drives will have keen interest in tape mount requests but will not be so concerned about interactive users logging on and off. A security specialist, on the other hand, should be aware of remote users logging on through dial-up modems. A tape librarian will profit from reports regarding the frequency in which particular tapes are used, and from knowing when these tapes are needed during the production schedule.

To design a proper operations management system you will need to identify the flow of information within your company: from users to operations staff and technical support, so that you are sure to have accounted for all types of messages. This will help you to identify the information which is critical to your operations and who should receive these messages. It is only when you have a good understanding of your environment that you can move towards centralization, the separation of tasks and automation.

Why Centralize?

For the same reasons that you want to divide console messages into logical categories for your operators, you should aim to structure your DP department in a manner that reflects the logical structure of your installation. In multi-machine environments, machines are rarely isolated from one another with respect to your company's production. Most shops network their machines in order to ease the transfer of information. There are usually many dependencies among machines for production, so the independent management of each HP3000 is an inefficient way to control your DP environment. It would be better to have control and management of multiple systems centralized so that all information can be easily related together. The result is that DP managers can make better management decisions and increase overall network uptime because they are working in a more controlled environment.

Some of you probably have, or are already taking steps towards centralization today. Batch scheduling software is a good example. It allows you to do sophisticated batch job scheduling and gives you the ability to have cross-machine job dependencies. Such utilities are available from third party vendors such as OCS and Unison. By knowing the status of jobs on all of your systems at one central site, you can truly run a distributed network efficiently. When provided with up-to-date information about the status of all your systems and by having all production scheduling information routed back to a central site, you can schedule jobs throughout a network and load balance across a number of systems. You can maximize the use of your resources by having all systems available from a central location.

Centralization also has implications on system security. By having central control of all consoles, you are reducing the number of access paths to your systems; that is, fewer people need the high level capabilities for operating your machines. By minimizing the paths for access to your systems, you increase security. Secondly, since all console messages are centralized, it is easier to watch for unauthorized access. Logon violations can be grouped together for easier identification.

Centralization of your consoles will also allow you to centralize your MIS staff: both operators and specialists. If all machines can be accessed and full console control is available from one location, then you don't need to have duplicate staff at remote sites. Because commands can be issued centrally, your resources and expertise can be at one site.

Towards Automation

Once centralization is achieved, you can start to consider automation. It is, however, important to note that automation is a relative term. It should be viewed as an effort to reduce human intervention without necessarily eliminating it. So, the real issue is how much to automate. Significant gains in reduced intervention can be attained without substantial costs associated with automation and, in particular, artificial intelligence applications. Also, automation comes with time: until you know exactly how and what to automate, you shouldn't try to do it.

Before implementing an automation strategy, you need to know exactly what to automate. The key ingredient for such an analysis is information: log all your operations activity, that is, the console traffic over a period of time. Next, analyze it to see what can be automated, design a plan and only then start to implement procedures. Undoubtedly, you would have gone through a similar process in implementing a job scheduling package.

In the instance of a job scheduling package, it's a tool that most shops "grow in to". In many cases, system managers start out by using the MPE :STREAM JOBxxx;AT=time command to run jobs at night unattended. Running jobs overnight is often the solution to completing processing without effecting performance for on-line users. However, where job dependencies exist and jobs have to be run sequentially, using the STREAM command may not be enough. With STREAM, you would have to guess when job A will complete so that you can stream job B to run once A has finished. If your predictions are off, jobs could be colliding or alternatively, you could have a lot of idle time on your system. At this point, a shop considers a job scheduling package to automate all overnight procedures. Once the scheduling package is installed, it is up to you to carefully plan your production so that processing continues properly. Typically, when people learn that the package is up and running, they will find more and more jobs that can be scheduled into it. Anything that can be automated, will be automated over time. Most shops find that automating job scheduling makes life easier for operators as there is less of a need for human intervention. The result is an increase in overall productivity of your machine since available processing time has been optimized.

In automating your entire operations, the process for implementation is much the same: review what you are currently doing, analyze what can be automated, plan for implementation, and implement. The job doesn't stop there either. Just as jobs continue to be added to a scheduling package, you should continue to evaluate tasks as candidates for automation. After the initial implementation, your focus becomes one of review and analysis of tasks to be automated.

Where Should Analysis Start?

Listing from your console will give you an accurate picture of what happens on a system from an operational point of view. From a log file, you need to identify which mechanical tasks it is possible to automate. In many cases, either HP or a third party offers a way of automating a task to varying degrees (as in the example of batch job scheduling). Currently, there is a wide variety of software packages that can be purchased to help you automate: automatic error detection, high-speed backup, database manipulation, spooling, performance analysis, and so on. An in-depth study of console activity and your overall operational goals will help you to decide which products may be of benefit to your shop. The key to these tools when moving towards automation is that they are programmable, where expert systems and artificial intelligence may be closest to complete automation.

As I discussed earlier, the most critical console messages are those that require human intervention. These could range anywhere from a reply to a tape request to correcting an error in a job to restarting a failed system. In each case, someone has to be made aware of the situation, otherwise, processing will not continue as planned. What is needed is some sort of alert mechanism. Depending on the criticality of the message, the appropriate reporting mechanism could require a console command to be issued, a message to be sent to a specialist, or a programmer on call to be contacted. Through an alert mechanism, you will know how much human interaction is necessary and where. This is invaluable information for your operations automation plan to help determine your requirements for staffing and software tools.

Simply stated, the rationale for analyzing your console activity, is so that you will know exactly what to automate and how. This helps to make sure that all bases are covered when moving to automation. The more information you have about your console activity, the more situations you can account for. Again, going back to the job scheduling example, if you know what the various types of possible errors are, then you can set up a number of contingency plans to compensate for those errors. Another example is system failures and automation. Through analysis of your console activity, you will know what are the most common reasons for system downtime, and you can then set up procedures to restart your system, tailoring the restart to the type of failure.

Also remember that automation is something that you can grow in to. In the early stages, the mechanism you use to alert personnel to critical situations may simply be a beeping message at the console. Later, it may grow to include a voice messaging system that automatically dials a programmer on call. To handle the problem of loading tapes and replying to requests, some companies are going so far as to get robots to mount tapes and then a reply can be programmed to execute automatically.

Even for less critical activity, it is a good idea to also log information so that you can monitor the demand for certain types of tasks. Again, this will also help your specialists to complete their jobs more efficiently. Take the example of system performance. Both Hewlett-Packard and Carolian Systems are taking steps towards making it easier for users to centralize and automate performance analysis through the creation of an interface between a performance tool and the console. HP has chosen to tie OPT/3000 to their recently announced OpenView product which will allow performance relation information to come back to a central site. Carolian has added a module to SYSVIEW, so that it can run in a "monitor" mode, which is designed to pass information about unusual performance situations back to a central console.

In summary, in order to manage multi-HP3000 sites effectively, centralization and automation of console activity are necessary. Careful analysis and logging of information is required to develop an effective operations optimization strategy. Once your objectives have been defined, you can start to implement your plans. Centralization of control of all HP3000's will increase your control, allow for more standardization and optimize the use of both hardware and human resources.

Automation should be implemented over time, so that you can automate as many different tasks as possible. Any tools that you choose to use to help you automate should be flexible, programmable and allow for growth. Automation frees up time for your operators so that they can concentrate on critical tasks. As utilities become more and more sophisticated, it may even be possible to program these tasks.

From reading through any of the HP publications, you will note that centralization and automation are the directions of many vendors and users alike. Many companies are realizing the need for standardization and efficient use of both hardware and human resources, and hence the interest in centralization and automation.

SALES FORCE AUTOMATION - THE LAST FRONTIER

BY MARK P. SHIRMAN - INNOVATIVE INFORMATION SYSTEMS
(IISI) NORWOOD, MA.
AND EDWARD F. LUCIA, JR. - IDSC RENTAL COMPANY INC.
MANCHESTER, N.H.

I. INTRODUCTION

Automated systems have become commonplace in almost every industry providing solutions in a variety of different application areas. There are scores of potential software alternatives for Accounts Receivable, General Ledger, Order Processing, and Inventory Control, but the whole area of automating the Sales Force is a concept that is just coming into its own. For many years salespeople have had to rely upon detailed recordkeeping and dogged determination to insure that sales prospects did not fall between the cracks. Additionally, management had very little way of accurately measuring the performance of marketing campaigns, telemarketing efforts, and their sales force as a whole. This paper will provide an introduction into the world of the fully automated sales force. We will discuss some basic philosophies as well as explore some of the features and functions that should be present in a sales lead management solution.

II. WHY AUTOMATE ?

There are literally dozens of reasons why a company should consider automating its sales force. Some of the major benefits are as follows:

- Provide a tool for the sales reps so that leads are appropriately and completely followedup
- Provide a tool for management to gain better control over not only the sales side of the business, but the entire business operation.
- Provides a beneficial Return-on-Investment
- System can provide an interface to the other automated applications in place
- Allows more comprehensive and detailed marketing analysis to be performed
- Provides a method to standardize the approach to sales, so that personnel turnover can be more easily managed
- Provide an easy way to import or handle mailing lists from outside sources as well as export information to other systems or companies.

The first of these benefits, a tool for the sales reps, can easily be put into perspective through the following graphic:

WHY DO CUSTOMERS STOP BUYING?

1%	Die
3%	Move Away
5%	Buy from a Friend
9%	Buy from Competition
14%	Product Dissatisfaction
68%	Because of no contact or Indifference or Attitude of Sales Rep

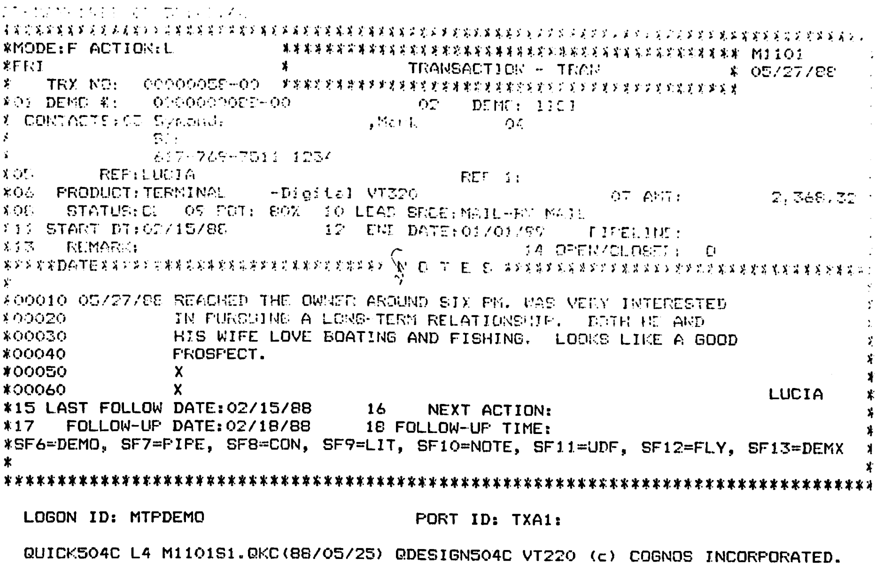
(Source: Edward Leader, Professional Sales Seminar 1981)

These are some pretty telling statistics. In short, the more leads that a rep can followup or touch on the more business they will generate. However, it can be an extremely difficult task to reach all the potential leads in an orderly manner, especially if the sales rep is already overburdened. An automated system can provide a rep with the tools to develop an organized approach to lead tracking.

Most good sales lead tracking software can generate tickler or action work files for reps to work off of. These files can be generated based upon followup dates, client interest, geographic location, product interest, and so on. This provides the rep with a more focused view of what they have to do for that day or week. The lists can be used to fuel either telemarketing operations or for personalized sales calls. The average cost for a sales rep to make a call in the U.S. is about $300. Based upon this fact alone, it would appear wise to be able to selectively followup and track those leads that are of the highest potential. The following page is an example of one of the central screens in on-line sales lead tracking package. You can see the types of information that can be maintained by the programs. Everything is here at the sales rep's fingertips to make an well-informed sales call either on the phone or in person.

How does management gauge the success of their sales force? The obvious answer is through the numbers of orders placed, however what they don't know is what percentage of business fell through the cracks or was generated from various marketing campaigns. Additionally, due to the generally independent nature of sales reps, it can be difficult to standardize the approach to lead tracking. This can become a serious problem when an individual leaves their job. Quite often valuable time and energy is lost as people try to figure out what the rep has left behind, if they did leave something behind.

An automated sales lead tracking package can assist organizations with many of the problems mentioned above. Most packages will be able to track the origin of a lead. This can be extremely helpful, especially in evaluating the success of a particular marketing/advertising campaign. The typical example would be an ad that generates a lot of "bingo card" type leads, or requests for product information. An automated system can help determine what percentage of those leads actually led to a sale as well as what the overall potential of these were. Additionally, lead tracking software has the ability to handle a large influx of leads that could be a result of a major campaign.



```
[illegible]
*******************************************************************************
*MODE:F ACTION:L          ****************************************** M1101   *
*FRI                      *             TRANSACTION - TRAN         * 05/27/88 *
*   TRX NO:  00000058-00  ******************************************          *
*01 DEMO #:   0000000088-00                02   DEMO: 1101                    *
* CONTACTS:03 Symonds                 ,Mark        04                         *
*             SI:                                                             *
*             617-769-7511 1234                                               *
*05      REF:LUCIA                        REF 1:                              *
*06  PRODUCT:TERMINAL     -Digital VT320                 07 AMT:     2,368.32 *
*08   STATUS:CL   09 PCT:  80%  10 LEAD SRCE:MAIL-BY MAIL                     *
*11 START DT:02/15/88         12  END DATE:01/01/99    PIPELINE:              *
*13   REMARK:                                  14 OPEN/CLOSED:  O             *
*****DATE************************** N O T E S *********************************
*                                                                             *
*00010 05/27/88 REACHED THE OWNER AROUND SIX PM. WAS VERY INTERESTED          *
*00020          IN PURSUING A LONG-TERM RELATIONSHIP.  BOTH HE AND            *
*00030          HIS WIFE LOVE BOATING AND FISHING.  LOOKS LIKE A GOOD         *
*00040          PROSPECT.                                                     *
*00050          X                                                             *
*00060          X                                                   LUCIA    *
*15 LAST FOLLOW DATE:02/15/88      16    NEXT ACTION:                         *
*17   FOLLOW-UP DATE:02/18/88      18 FOLLOW-UP TIME:                         *
*SF6=DEMO, SF7=PIPE, SF8=CON, SF9=LIT, SF10=NOTE, SF11=UDF, SF12=FLY, SF13=DEMX *
*                                                                             *
*******************************************************************************

  LOGON ID: MTPDEMO                    PORT ID: TXA1:

  QUICK504C L4 M1101S1.QKC(88/05/25) QDESIGN504C VT220 (c) COGNOS INCORPORATED.
```

Management should view sales force automation as a tool, in the same manner the sales rep does. By having access to all the leads for an organization, management can accurately create forecasts, measure rep performance, and standardize its approach to lead followup. The key here is the increased level of control automating the sales force can bring to management. In this way it is no different than automating a manufacturing facility. Organizations often pursue this path in an effort to control costs and analyze more data, and it is no different with sales lead tracking systems.

Management should also be impressed with the return-on-investment figures that usually correspond to automating the lead tracking functions. The following pages represent some models that can be used to evaluate the ROI figures. We will start small and think about a company with annual sales of one million dollars and a software package that costs $14,400. You can see that even at a 5% increase in business the ROI figures are amazing. That 5% figure is much lower than many statistical surveys have shown. In fact, Sales and Marketing magazine found productivity increases averaged 43%(Summer 1987) with the introduction of an automated sales lead tracking system. The ROI figures coupled with the benefits mentioned before make a pretty compelling argument in favor of automating the sales force.

III. Philosophies and Approach

There is a school of thought that believes that sales force automation is best achieved through the use of microcomputers. We do not believe this to be the case especially in the HP environment. A centralized IMAGE data base allows sharing of information and files much easier than with PC's. One of the goals of sales force automation is that of management control. By having a central receptacle for data, this is more easily achieved. The central data base concept can also have implications as to how the system can interface with other automated applications. MIS should not view the sales lead tracking data as simply an "island of information". It is possible to feed order entry as well as accounting systems, and manufacturing systems the information generated by an automated sales tool. Conversely, the sales software should be able to "import" information from other data base applications. This entire concept will allow easier generation of management reports and development of inquiry subsystems.

Eventhough we advocate a centralized data base approach to sales lead software, the package should have the ability to "export" information to other processing options, ie. another HP3000, Vectra, or Portable. The concept can be thought of distributed processing with an interface to the host. Lets consider an example. A sales rep is in the field. He/she downloads a list of leads from the corporate office to a PC. The leads are managed, tracked, changed, and more are added. At the end of the day, week, month or whatever they upload those leads to the corporate office. The leads can still be maintained on the remote workstation, but the home office can evaluated them for forecasting and other analysis.

Return on Investment Example

per $000 in Annual Sales Volume
(Manufacturer Profile Assumptions)

	Current Base Amount	(Choose one below) <----Increase in Sales Due to----> Automation				
		5%	10%	20%	30%	40%
Sales	$1,000	$1,050	$1,100	$1,200	$1,300	$1,400
Manf. Cost of Goods	500	525	550	600	650	700
Gross Profit (50%)	$ 500	$ 525	$ 550	$ 600	$ 650	$ 700
Selling Expenses (20%)**	200	210	220	240	260	280
General & Admin. (Fixed)	200	200	200	200	200	200
Net Profit before Taxes	$ 100	$ 115	$ 130	$ 160	$ 190	$ 220
Net Increase over base	---	15	30	60	90	110
Percentage Increase over base profit due to Software	---	15%	30%	60%	90%	110%

** Worst case assuming all selling costs increase as the sales productivity increases. In actuality, some of these expenses will remain fixed at a lower level.

Return on Investment Example

Software Calculation Examples

(Manufacturer Profile Assumptions)

	<----Increase in Sales Due to Automation---->				
	5%	10%	20%	30%	40%
Case: $1,000,000 Annual Sales					
1. Increase in annual profit	15,000	30,000	60,000	90,000	110,000
2. Annual suuport fees	2,400	2,400	2,400	2,400	2,400
3. Software	14,400	14,400	14,400	14,400	14,000
(1. minus 2.) divided by 3. % = RIO					
	90%	191%	400%	608%	747%
3. divided by (1. minus 2.) times 12 = Payback in months	14	6	3	2	1

IV. APPROACH

As with any other system, the software functionality should be the primary driver as to what system should be put together. The functionality of sales lead tracking software is different from many systems that may go into a data center. It is important that the projects aimed at decisions of what packages to implement, incorporate the sales force in their process. Because sales lead tracking is a relatively new concept, not many MIS people understand all the ins and outs. It is currently a speciality unto its own. Before implementing a new system make sure that all the bases have been covered with an individual that knows these types of systems.

The functionality that needs to be present in sales force software is going to vary from organization to organization. There are some basic pieces that should however be present in every on of those systems. Most importantly the sysyem must be easy for the sales rep to use. This implies more than just user-friendliness. Flexibility, speed, and as few keystrokes as possible are all vital components of this. If the system is not easy to use, you will never get your sales reps to use it. The more the reps use the system the better sales results will be and the more the overall data will be helpful to management.

Some additional features/functions that should be present in an sales lead tracking package are:

- Ability to import mailing lists from third party or outside sources
- Creation of tickler/action fields to zoom in on only relevant data
- Ability to store free-form text easily along with lead information
- Ability to perform literature fulfillment and mailing labels for mess mailings
- Should be able to generate batch letters as well as customized letters on the fly
- Should have the ability to look at data a variety of different ways through keys
- Ad-hoc reporting features
- Flexible security so that there is no problem with reps stealing information from one another, but management can view all relevant information
- Ease of interface into other existing data processing applications

These are just a few of the major things to look for in a sales automation package. There are many more things which could be of importance to your own shop i.e., interface with graphics, audit trails, data dictionary, documentation...Just remember flexibility and ease of use are of primary importance.

Conclusion

It is our belief that the time to automate your sales force is now. Those organizations that do will have a competetive edge over their rivals. Most of the other major functional areas in an organization have been automated, the sales force and lead tracking areas represent one of the last frontiers in data processing. The ability to get a handle in an organized way of how your reps are performing, while at the same time giving your sales force a powerful tool, is a temptation all MIS directors should take seriously.

HP ThinLan - A Users View

Allen R. Burns
Rutgers - The State University
311 North Fifth Street
Camden N.J. 08102

The purpose of this paper is inform others of our experiences using HP ThinLan and an HP 3000 Series 68 as the server or host. The series 68 is networked to 25 IBM PCs in our micro lab. The paper will hopefully not be too technical but discuss how the network has been integrated into our computer services. It will cover the installation, management, advantages, disadvantages and some problems we have encountered primarily from the HP 3000 Series 68 side. It will address the configurations changes made to the Series 68, the performance of the Series 68 and the performance of the PCs.

Being a University we have a diverse set of computing needs. Our department provides support for academic computing done on the campus. The need of a local area network became apparent when our micro lab continued to expand. We were experiencing a decline in the use of the HP 3000 and an increase in demand for micro computing power and equipment. We had several PCs and wanted to be able to share resources as well as data between them. We still wanted to be able to use the PCs as stand alone machines and also as terminals connected to the HP 3000. A Lan would provide these features and the sharing of resources such as printers and plotters we desired. HP ThinLan satisfied these needs and using the HP 3000 as the host also gave us the ability to backup all the PC files as we were doing our weekly and daily dumps. We also gained the mass storage capability of our disc drives attached to the Series 68. HP ThinLan allowed us to make better use of the resources of our HP 3000, provide LAN services to our micro lab users, and have the PCs act as terminals for the HP 3000.

HP was helpful guiding us in purchasing all the hardware and software that we would need to network our lab. Installation was done by an HP SE (Jim Gagliardi) and CE (Roger Brooks) on one long Friday afternoon. The HP 3000 had to be shutdown for the installation of the hardware, software and configuration changes. The HP 3000 had to have a local area network interface controller(LANIC) installed on the backplane of the machine. The LANIC is a microprocessor-based communications controller and is the link point for the HP 3000. The loading of the software and the configuration changes to the 3000 were done by the SE. The customer has the responsibility of running the actual ThinLan coaxial cable between the host and the PCs. There was a joint effort installing the hardware and software on the PCs. It does require that the PC case be removed and a card installed. A boot disk had to be created for each PC also.

Each PC requires an HP ThinLan server kit which consists of an interface card, BNC "T" connector and user software for the designated PC. The HP 3000 requires the following products: ThinLan/3000 link, UB-Delta-1 or later, Resource Sharing software, a LANIC, and a ThinMau(Thin Medium Attachment Unit) that connects the Lanic to the ThinLan cable. The installation of the Resource Sharing software was done from a tape by the SE. The configuration changes were required to identify an LDEV for the LANIC, virtual terminals were created to handle incoming data communication connections, and some new groups and accounts were required, as our system did not have the HP Office account. There are guidelines that you are to use to calculate values for some table entries on the system. The tables values on our system did require some changes.

In the Resource Sharing: Systems Management manual (32597-90001) there are a few chapters on preplanning your network. There are three people designated who should have input in the planning process. The System Manager, the HP 3000 Network Manager, and the PC Network Manager should decide how many PCs are to be connected as workstations, the printers that will be available to the network, and any other devices that you may want to share. The current table sizes must be evaluated to see that they are large enough to handle the addition of the network. This is one area where you should spend considerable time. We had a plan of what we wanted to connect and it appeared that our table sizes would be able to handle the increased demands of the network. I did miss one sentence regarding the increase demand for Global Rins. I also suggest that the original planning be set up for some expansion. We initially planned to connect 10 PCs. When we reviewed the table sizes we were within range for the 10. An additional 15 PCs were going to be added at a later date. We should have planned everything to be configured to handle 30 PCs. This would have given us the room to grow and still have a cushion beyond our maximum number of workstations. There is no stated limit to the number of active workstations that are attached to the network when the HP 3000 is used as the server. There is a limit to the number of nodes attached to the network. We were told that a ThinLAN cable segment can have a maximum length of 185 meters and that no more than 30 nodes can be placed on one segment. This can be expanded to three segments with repeaters. Using this as a guide any one Thinlan network could accommodate close to 90 PC workstations. There are some limits that you should be made aware of if you plan to use a "PC" as a server. These limits have increased and probably will change again before too long. You will probably want to keep a close watch on the use of the table values by the Tuner program or other system performance package as the network begins to be used. We saw a slow beginning followed by a rapid increase of use as our users became aware of the advantages of the network. The heavier the use of the network, the more resources will be required.

There are worksheets that are found in the Resource Sharing: System Management manual to help in planning your network. I strongly suggest that you complete the forms. This will take a lot of time but, will guide you in designing the structure of the network and will help your HP SE in configuring the system. Again give yourself some growing room. The amount of resources that you are setting up on the HP 3000 for the network should not cause a problem. We did expand the table values as our network use grew. Our user became dependent on the network, especially when they knew the default printer was a LaserJet Series II which was upgraded in the early spring to a Laser 2000. We were happy to see the use increase and moved up our addition of the other 15 workstations. We had originally planned to run the network on a Zenith 80386 based machine as the server. Using the Zenith as the server would have limited us to only 10 active users on the network for our version of Resource Sharing. The network still remains on the HP 3000 as we have been happy with the operation of the network in general. We still plan to place the Zenith machine onto the network as a server.

We have obtained the essential advantages we were looking for from a LAN and some that we feel are nice options. The network has allowed us to share printers and data between machines. With the network connected to the HP 3000, we also have gained terminals that we did not have before. Students and faculty can now upload and download files very easily. Students and faculty can backup their files that they have used during the semester on floppy diskettes or convert MPE files to PC files and vise-versa. All files are backed up daily and weekly with our regularly scheduled HP 3000 dumps. We have the mass storage capability of the HP 3000. Public domain software, shareware, and commercial software are readily available to the users of the network from a drive that is as easily accessible as an internal drive on the PC. If you intend to put commercial software on you network, request the network version. You will have problems if you do not have the network version and you will comply with the copyright laws of the products. A site license is another way to comply with protected software. The network makes it easy to insure that all parties are using the same version of software. It also helps when there is an update to a product. In our case, it reduced the number of lost and damaged diskettes.

There are some disadvantages that go with the network. The performance of both the HP 3000 and the PC will be discussed shortly. I will confine my remarks here to nonperformance types of problems. On our current version of the Resource Sharing and OfficeShare software there are some restrictions that are being addressed by later revisions. The limit of 10 active workstations on the network, if a PC is used as the host, was a problem for us. This was a factor in our choice to stay with the HP 3000 as the host. This limit is supposed to be increased in the performance version of the software. There is the limit on the length of cable on the network which has not caused us any problems. Network software costs may be a problem if you have to purchase enough copies of a specific product to comply with the copyright laws in cases where a network version is not available. The manuals are not bad for a young product, but they could be a little more comprehensive. I dislike the reference to another manual that you usually did not purchase. It would also be nice if they would refer to a particular section or chapter in the other manual instead of the whole manual. I did purchase one additional manual; NS3000/V Network Manager Reference Manual, as it is referred to quite often. Our backups are done the first thing in the morning and the network is down during that time which causes some users problems. There is a limit of 1200 files on a shared disc that has caused our PC Network Manager difficulties. We have also noticed an increase in the number of files dumped daily. The network files are dumped if they are accessed.

Performance of the network has to be viewed from both the HP 3000 side as well as the PC side. As I said before, we were experiencing a decline in the use of the HP 3000 Series 68. We had the resources available to spare and the addition of the network has actually been a plus for our situation. We were able to use an existing resource to improve our overall services. I will admit that having the network has affected the response times of the HP 3000. This is especially true when a significant number of printouts are being routed over the network and when a PC is being booted onto the network. We boot the PCs on the network when the lab is opened in the morning and leave them connected unless there is an individual who wants to use the PC as a stand alone. The boot time to attach a PC to the network is long. This time averages less than 3 minutes for most of our machines, which usually occurs once a day. Applications are downloaded from the HP in the same way you would load an application from any hard or floppy drive attached to the PC. Since the LAN is transparent to the user, they do not have to learn new commands. This load time depends on the software package and overall system use. Once the package is downloaded there is a slight noticeable difference in its speed of operation if it has to access the network disk for more information. Our users have not complained about this slight decrease in response time. The initial download time is less than the time it would take the user to signout the package from the aid station and load it. The user does not have to wait in line for the aide to sign out the package or return it. Printing does take longer over the network. The files are printed in another room across the hall on a laserjet 2000, that allows us to monitor the printing. Since we give our users high quality printing, we have not had any complaints about the time it takes to print or retrieve it from across the hall. We could not justify a laser printer attached to each of the 25 PCs so we have no complaints. File transfer between the PCs and the HP 3000 on the network is much faster than any other software communications package that we have.

Managing the system is done by a series of programs in the HP Resource Sharing product. Resource Sharing allows PC users in the office environment to share and save PC files on the HP 3000. It allows for printing of documents on the HP 3000 printers and backing up of PC files. Resource Sharing works with another HP product OfficeShare that is the PC side software. OfficeShare links PC users on a network. Resource Sharing also permits a PC user to create shared discs files on the HP 3000, to use shared discs as extensions of their PC discs and allows the conversion of DOS format files to MPE format and vice versa. Resource Sharing has a function key and command mode interface. Online help is available for most screens. The HP 3000 Network Manager uses Resource Sharing to create the PC shared discs on the HP. Shared discs on the HP 3000 have a structure similar to DOS tree structures with directories and pathnames. Access to files can be restricted to PC users and the files can have read/write/create attributes as well. There are 5 areas that you control or manage from Resource Sharing. Typing (:RUN RESMGR.PPC.SYS;LIB=G) will execute the program to enter this environment. Disc functions allow you to create shared disc and control access to these shared discs by other users on the network. Configuration functions control the setup of the network parameters. System functions give the Network Manager control of the Resource Sharing table configuration files and the creation/deletion of shared discs. One more feature is the Diagnostic functions to monitor the network for problems or potential problems. The use of these functions is limited by a users capabilities - SM (system manager), AM (account manager) and NM (network manager) are necessary to use many of the functions.

Files may be backed up 2 ways on the network. The easiest way is the daily and weekly dumps done on the series 68. There is also a BACKUP command in Resource Sharing that allows you to dump only the network files. These files can be recovered from either backup method by using the Recover command in Resource Sharing.

The file convert utility (:RUN FILECONV.PPC.SYS;LIB=G;NOCB) provides the user with the ability to convert from the PC DOS format to the standard MPE format or vice versa. This utility also allows you to display your DOS directories and copy shared disc files.

A Print Spooler Utility is used to manage the network printing. This utility system (:RUN PSUTIL.PUB.SYS) allows users to monitor and control the print spooler. PSUTIL is used to design the printing environment for the shared network printer devices. The print spooler supports both large and small printers, controls spool files, manages attended printers,and can recover or reprint spool files. The spooling queues can be viewed and priorities of spooled files changed. PSUTIL is also used to add, modify or delete print devices that are to be shared on the network. PSUTIL can be run in command mode or by function keys. The System Manager uses PSUTIL to start and stop the print spooler for various reasons such as backups. There is on line help available. We have two printers configured on the network and both are unattended. Once you have the printer devices configured you generally do not have to do too much with PSUTIL unless you have attended printers on the network. Attended printers require the supervision of an operator to respond to print request issued by the PC users.

The last program that I will take a quick look at is the NMMGR program(:RUN NMMGR.PUB.SYS). This program is required to be run the first time Resource sharing or NS/3000 has been installed on your system. I ordered the NS3000/V Network Manager Reference Manual (32344-90002) to have more documentation of what was happening with the network, since we do not pay for RCS support. The Resource Sharing System Management manual does refer to this manual and not having access to an SE or the response center I felt I had to have something more to rely upon when I had questions. The program has online help but is not as well documented with online help as some of the others. NMMGR enables the SM or NM to view more of the configuration files on the network. If you have an SE, you would probably want him to show you more about this program. I have not injured the system by poking around with it. The manual does explain the purposes of the network config files, catenets, bridges, and gateways. It is very interesting reading.

We are very satisfied with the HP ThinLan Network. The product has performed well. There are some problems as I have mentioned but, the advantages we have gained are significant. Our users like the availability of the software and printing. I like the ease of managing the product. Our PC Network manager likes the file storage capabilities of the network. We have less problems with diskettes being ruined or disappearing. The network has been reliable. We now have more workstations that can access our HP 3000. Use of the HP 3000 has increased. With all the benefits, we are willing to suffer some performance problems. HP already has released a performance version of the network software for UB-Delta-4 and up. Some of the performance problems have been resolved by the new release. HP has a good product in ThinLan.

Reducing Migrating Secondaries

Earl Waller and Mel Pearce
INLEX, Inc.
P.O. Box 1349
Monterey, CA. 93942

If you are like us, you have read a lot recently about IMAGE and its 'myths', and you know that there can be performance problems associated with migrating secondaries in an IMAGE database. And like us, you follow the recommendations to use ASCII key structures and prime number capacities. And, finally, you expect and accept some performance degradation as the database fills up and schedule your database reloads over long weekends. But what do you do when the performance becomes unacceptable?

INLEX provides library automation solutions. Design considerations led us to use a combination of MPE flat files (accessed randomly via pointers), KSAM files (for partial key searches and sorted access) and IMAGE databases (to give us fast known key, multi-record access). As relative newcomers to IMAGE, we read all of the available information about IMAGE database design and wanted to follow all of the recommendations.

Due to several restraints, we found that we needed to use a binary pointer as a key. We discussed the problem with an IMAGE expert who suggested that we use a six character ASCII key as the binary pointer but store it as a binary value. He thought that this might give better results than using a three word binary key.

Our customers routinely experienced performance variability when loading bibliographic records. A typical bibliographic record requires two variable length flat file records, eleven KSAM records, and three IMAGE records. Load rates differed across our customer base from one to thirty records per minute and degraded severely as the databases filled up. This seemed to follow very closely the bibliographic record load rates of other vendors in our industry and, because we had been told that KSAM was very slow when rebalancing its trees, we didn't believe that migrating secondaries were a serious issue.

The real problem surfaced when we tried reloading one of our customer's IMAGE databases. Using a dedicated Series 52 computer, we initiated a database reload of an IMAGE database containing 141,000 records. We aborted the not yet completed computer run after two and a half weeks of dedicated time with the realization that we had some serious migrating secondary issues to resolve.

About this time there were numerous articles in the literature exposing myths about prime number capacities and integer keys. However, these articles failed to explain how to analyze a database, how to tune a database for performance, or even how to know that the database is optimum.

Once we were confronted with the problem we realized that we needed an analysis tool and a methodology to answer these questions. We searched for tools to combat the problem but were unable to find anything that promised sufficient insight or a clear solution. So, we decided to build our own tools.

We obtained the IMAGE hashing algorithms and implemented a program that would read an IMAGE master and chart the distribution of the hashed keys. The program also gave a measurement of the number of primaries with no synonyms, with one synonym, with two synonyms, etc.

Needing more disk to hold the customer's database for testing and realizing we would be making hundreds of disk intensive runs, we obtained a 132 megabyte ram disk from Imperial Technologies[1]. We were able to configure the device as an HP 7914 disk drive, allowing us to perform high speed disk i/o, eliminating seek and latency time.

With our customer's 141,000 record database and our new program, we cycled through the automatic master hundreds of times using hundreds of different capacities. Because the program only totaled secondaries and did not migrate them, it ran in a fraction of the time needed to reload a database. The program generated the calculated hash distribution for each run and displayed the results in a series of bar charts. Figure 1, on the next page, is an example of one of these bar charts.

Analysis of the charts showed that many of the keys hashed to the same address and/or the same general area in the automatic master. This created clusters and synonym chains, many of them quite long. When IMAGE computes a synonym it adds the entry to the end of a synonym chain or creates a new chain and attempts to place the new entry in an empty slot in the same block of the file. When the existing block is full, IMAGE places the new entry in another block, causing an overflow. The computed overflow block count showed that the empty slots in the hashed block were filling up and that IMAGE placed the entries in additional blocks, sometimes after very long serial searches for an empty slot.

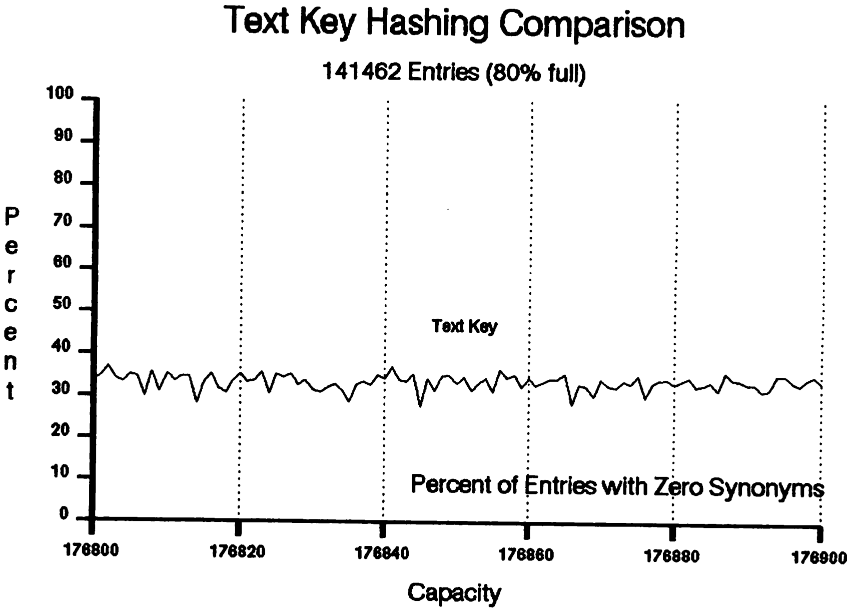
Figure 1 - Sample Bar Chart Report

```
                      Data Base = AUTHOR.DBS
No.  Name           Type  BF   Entries   Capacity
1    AUTHOR-KEY      A    28   141462     324001
                 Data set number = 1
                     Search item = KEY-POINTER
                       Item type = X
                         Capacity = 190973
                 Blocking factor = 28
                     Entry count = 141462 (74.1%)
        Entries with  0 synonyms -       50960   36.0%
        Entries with  1 synonyms -       33660   23.8%
        Entries with  2 synonyms -       17979   12.7%
        Entries with  3 synonyms -       12708    9.0%
        Entries with  4 synonyms -        9800    6.9%
        Entries with  5 synonyms -        7404    5.2%
        Entries with  6 synonyms -        4347    3.1%
        Entries with  7 synonyms -        2448    1.7%
        Entries with  8 synonyms -        1197    0.8%
        Entries with  9 synonyms -         640    0.5%
        Entries with 10 synonyms -         209    0.1%
        Entries with 11 synonyms -          96    0.1%
        Entries with 12 synonyms -           0    0.0%
        Entries with 13 synonyms -          14    0.0%
             Overflow block count = 729
                Total block count = 6821

17780+ *
     | *
     |**
     |**
13335+**
     |**
     |****
     |****
 8890+****
     |****
     |****
     |*****
 4445+*****                  *
     |******      **         **
     |*************** ***********    * ****************
     |**************************************************
     +----+----+----+----+----+----+----+----+----+----+
         10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
```

When IMAGE hashes to a slot occupied by a synonym, it moves the existing synonym entry into another empty slot, adjusts the synonym chain pointers, and places the new entry into the vacated slot. This is called migrating a secondary. It also can be very disk intensive.

The IMAGE hashing algorithm is a function of two variables: the key type and the master dataset capacity. We modified our analysis tool so that we could analyze up to fifty different capacities in one computer run and summarize the result into a single table. The results of this series of tests showed that prime numbers are not necessarily the best capacities to choose. Now we could see the best capacities for the data in the databases. However, the occurrence of secondaries in the best selection was still larger than we could accept. Figure 2 shows that, for these tests, the number of entries with zero synonyms was not even equal to forty percent.

Figure 2



Our next step was to analyze several alternate key structures. To do this, we created another program that would accept input from a flat file and perform the same functions as the first program. This allowed us to analyze potential key structures without first building and loading a database.

The keys of concern are the pointers to the file and a logical record within the file. In accordance with IMAGE recommendations, we originally specified this to be an alpha key. To put it another way, we purposely avoided using an integer key.

We chose an integer key for our next series of analysis runs. Figures 3 and 4 show a performance comparison of the use of an integer key versus the use of a text key over a broad range of capacities. We observed that the integer key resulted in significantly improved distribution, and smaller clusters, with a significant reduction in the number of synonyms and the length of synonym chains.

Figure 3

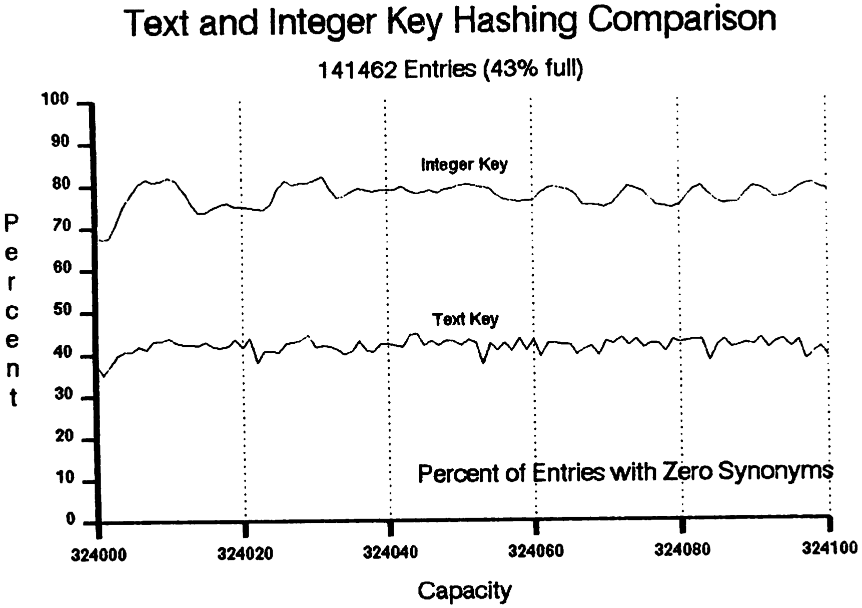
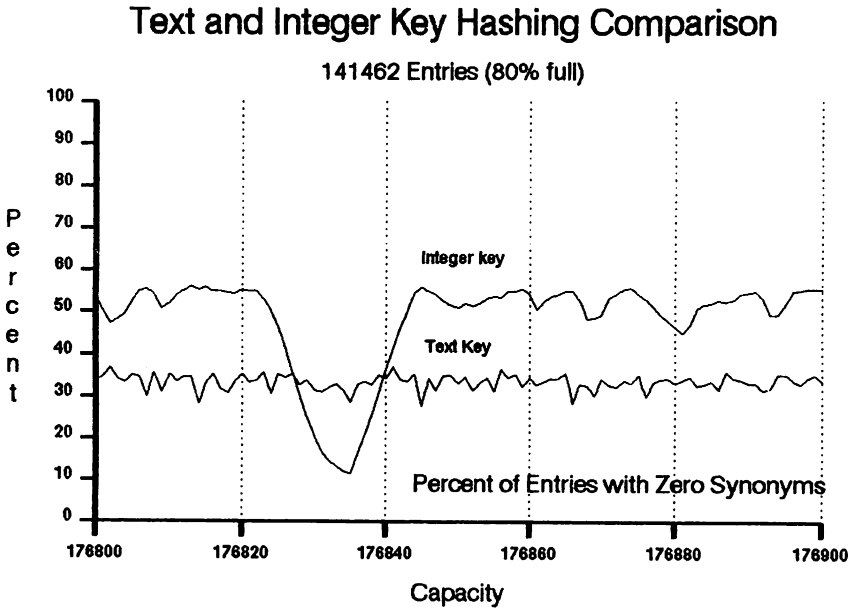


Figure 4

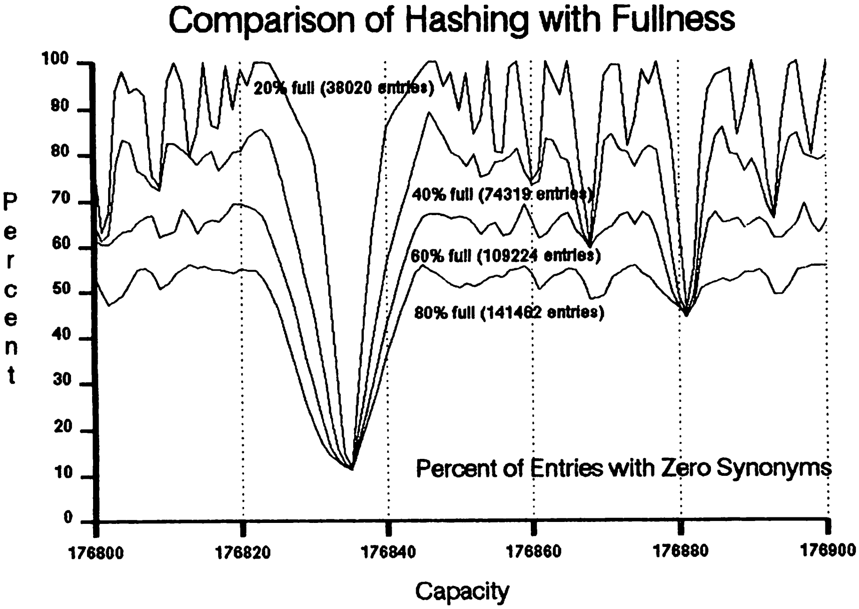


We felt that we were on the right track; however, the computer time required for the analyses, even with the ram disk, was significant on our Series 40 computer.

We wondered if we could derive any conclusions from comparing smaller chunks of the same data in the same database for a range of capacities of the master set. This caused us to make a number of runs using only enough data to fill the master to twenty percent, forty percent, sixty percent and eighty percent of its capacity.

Figure 5, on the next page, shows the analyses generated for some of these runs. We observed that goodness for a twenty percent full database never translates into badness as the database becomes full. Our conclusion was that poor capacities could be avoided by analyzing a sample as small as twenty percent of the data.

Figure 5



By now we had developed some confidence in our analysis tool. The real measure of its success, of course, could only be determined by measuring the results in the real world.

We selected the integer key and an optimum capacity and reloaded the database at our customer site. The reload that previously had not completed in two and a half weeks finished in just five hours. Encouraged by these results, we were anxious to see how these improvements would affect our database loading process.

We selected a customer with slightly more than 208,000 bibliographic records to load. The customer was using a dedicated HP3000 Series 70 with Eagle drives. Load times of eight to ten records per minute were originally anticipated. Based on a 20 hour day of productive work (time off for system backup) it would take approximately 28-29 days to load and index this database.

All 208,000 plus bibliographic records were loaded and indexed over a three and one quarter day period. It was encouraging to note no observable performance degradation as the databases filled.

We are now using the tools we developed prior to any database reloads, and any time there is a design change to an IMAGE key structure.

As graduates of the school of hard knocks, we have formed some conclusions. It is possible to tune for optimum performance and know when the optimum is reached. Prime numbers may or may not be good choices for a dataset capacity. Two identical databases can require different optimum capacities if they contain different data values. Most important of all, performance and tuning is a function of the specific data in the database, and without a tool to examine a database there is no way to know what IMAGE is really doing with your data.

1. Imperial Technology, Inc.
831 S. Douglas Street, Suite 102
El Segundo, CA 90245
(213) 536-0018

CREATING BANNER PAGES

Edmund C. Arranga
Business Programmer/Analyst
McDonnell Douglas Corp.
3855 Lakewood Boulevard
Long Beach, CA 90846

INTRODUCTION

This paper describes for the HP programmer a process to create banner pages for reports. The banner pages will contain report distribution information specific to the reports, yet remain independent of the reports' creation. The process to create report specific banner pages with mailing information is divided into 3 major steps or components. These components are MPE commands, SPOOK, and COBOL. Each component in the process will be examined and explained individually before the entire process is presented.

I.) MPE COMMANDS (Multiprogramming Executive operating system)

The banner page creation process uses 6 MPE commands. They are: BUILD, FILE, HEADOFF, HEADON, OUTFENCE, and SHOWOUT. For reference, the exact syntax used by the 6 MPE commands is given below.

```
1.) :BUILD SO;REC=-80,,F,ASCII;DISC=100 * :BUILD SI;REC=-80,,F,ASCII;DISC=10
2.) :FILE BANNER;DEV=137,  :FILE BPAGE;REC=-133,,F,ASCII;DEV=137,1,1
3.) :HEADOFF 137
4.) :HEADON 137
5.) :OUTFENCE 12;LDEV=137
6.) :SHOWOUT JOB=@:DEV=137
```

SHOWOUT

The MPE SHOWOUT command reports the status of output device files. These are the files or reports waiting to print. To illustrate, the SHOWOUT command is demonstrated below.

```
:SHOWOUT JOB=@;DEV=137

DEV/CL    DFID     JOBNUM  FNAME    STATE FRM SPACE RANK PRI #C
137       #074     #J'133  MAILINFO READY          40     D 1  1
137       #075     #J'294  REPORT1  READY         252     D 1  1
```

*The HP 3000 prompts are used to indicate specific levels of user interaction.
The colon (:) indicates the operating system level
The greater than sign (>) indicates an interactive SPOOK session
Example MPE commands differ slightly from the syntax of the above commands

```
2 FILES (DISPLAYED):
   0 ACTIVE
   2 READY; INCLUDING 2 SPOOFLES,  2 DEFERRED
   0 OPENED; INCLUDING 0 SPOOFLES
   0 LOCKED; INCLUDING 0 SPOOFLES
   2 SPOOFLES:   272 SECTORS
OUTFENCE = 1
```

In this example two output spool files, MAILINFO and REPORT1 are waiting to print on device 137. Their DFID's (device file ID) are #074, and #075 respectively. The FNAME and DFID will be used to uniquely identify output spool files to tag with report distribution information.

For demonstration purposes it is assumed that MAILINFO contains report distribution information specific to REPORT1. If the two files are joined the result would be a report with a banner that contained distribution information. The ability to join these two files is examined next in the section on SPOOK.

II.) SPOOK

The second component in the banner creation process uses SPOOK. SPOOK.PUB.SYS is a utility program supplied with MPE which allows the user to interrogate and manipulate spool files. Different versions of MPE run different versions of SPOOK. Nonetheless, the command sets are identical for SPOOK, SPOOK4 and SPOOK5. This paper will use the term SPOOK as a generic reference to all three versions.

SPOOK's APPEND COMMAND

The SPOOK APPEND command, "APPENDS PART OR ALL OF AN OUTPUT SPOOL FILE TO ANOTHER OUTPUT SPOOL FILE." Herein lies the key to creating banner pages. A report waiting to print (an output spool file) can be appended to another report (another output spool file) waiting to print.

Before the APPEND command is demonstrated SPOOK has two other useful capabilities which will be briefly discussed. The first is, the ability to use MPE commands while running SPOOK. In fact, any MPE command that can be accessed programmatically by the COMMAND intrinsic can be accessed by SPOOK. The second, is the ability to back reference a file equation in the APPEND command.

These capabilities along with the APPEND command are illustrated below.

```
:RUN SPOOK.PUB.SYS

SPOOK G.02.B0   (C) HEWLETT-PACKARD CO., 1983
>FILE BANNER;DEV=137,12,1
>APPEND #074,#075;ALL,*BANNER
>APPEND END
```

```
>SHOWOUT JOB=@;DEV=137

DEV/CL    DFID     JOBNUM  FNAME    STATE FRM SPACE RANK PRI #C
137       #076     #S1076  BANNER   ACTIVE      272    1 12  1
137       #074     #J'133  MAILINFO READY        40    D 1   1
137       #075     #J'294  REPORT1  READY       252    D 1   1


3 FILES (DISPLAYED):
   1 ACTIVE
   3 READY; INCLUDING 3 SPOOFLES,  2 DEFERRED
   0 OPENED; INCLUDING 0 SPOOFLES
   0 LOCKED; INCLUDING 0 SPOOFLES
   3 SPOOFLES:   564 SECTORS
OUTFENCE = 1
```

MAILINFO and REPORT1 have been appended to create a new spool file, BANNER with the parameters assigned in the file equation. BANNER is 'ACTIVE' (printing) on device 137 because of the output priority assigned in the file equation.

REDIRECTING SPOOK's STDLIST and STDIN

Until now the information returned by the SHOWOUT command has been directed to $STDLIST, or the terminal. To capture the same output spool file information in a more permanent manner, STDLIST can be redirected to a disc file. First the BUILD command is used to construct a MPE file.

```
:BUILD SO;REC=-80,,F,ASCII;DISC=100
```

This permanent disc file, SO is where SPOOK will be directed to return information.

```
:RUN SPOOK.PUB.SYS;STDLIST=SO
```

MPE now runs SPOOK but the terminal does not display the familiar SPOOK signon. Instead this information has been sent to the file SO. Continuing, the SHOWOUT command is entered;

```
SHOWOUT JOB=@;DEV=137      (again SO captures the information)
EXIT                       (terminate SPOOK)
:                          (return to the operating system)
```

At this point, the ability to redirect STDLIST, and the ability to use MPE commands within SPOOK has allowed the programmer to capture output spool file information on a permanent disc file.

In the same manner SPOOK is directed to send information to a disc file, SPOOK can be directed to receive commands from a disc file. In a previous example the SPOOK APPEND command was used to create a new spool file. The same commands used then interactively, now can be directed to SPOOK from a disc file. The disc file, SI contains the information as follows:

```
FILE BANNER;DEV=137,12,1
APPEND #074, #075;ALL,*BANNER
APPEND END
EXIT
```

SPOOK is run now with STDIN equated to SI, the command file.

```
:RUN SPOOK.PUB.SYS;STDIN=SI
```

SPOOK signs on, looks to SI for its commands, carries out the commands and signs off, per the EXIT command.

CALLING SPOOK PROGRAMMATICALLY

This section introduces a small SPL (systems programming language) program that calls SPOOK. (NOTE - SPL is not mandatory to create banner pages. All the commands used in the SPL program can be coded using a higher level language such as Pascal or COBOL. The choice is strictly up to the programmer. By using SPL however, the programmer avoids the wordiness of COBOL and has available a small utility program useful in other applications).

The program listed below named 'CALLSPK' equates STDIN to the command file SI. A COBOL program introduced in the next section will be responsible to build SI with the appropriate APPEND and SHOWOUT commands. The file SO is equated to STDLIST and is used to capture the information generated by the SHOWOUT command.

```
$CONTROL SUBPROGRAM
BEGIN
PROCEDURE CALLSPK;
BEGIN
INTRINSIC CREATEPROCESS, ACTIVATE, KILL;
OWN ARRAY B (0:2);
OWN INTEGER ARRAY IA (0:2) :=1(9,8,0);
OWN LOGICAL ARRAY LA (0:1);
OWN BYTE ARRAY BA1 (0:39) := 1("SO.PUB.PRODW",%15);   equate STDLIST to SO
OWN BYTE ARRAY BA2 (0:39) := 1("SI.PUB.PRODW',%15);   equate STDIN to SI
OWN BYTE ARRAY PRINTPROC (0:39):= "SPOOK.PUB.SYS";    call SPOOK
INTEGER ERROR,PIN;
LA(0) := @BA1;
LA(1) := @BA2;
CREATEPROCESS (ERROR,PIN,PRINTPROC,IA,LA);
ACTIVATE (PIN,2);
KILL (PIN);
END;
END.
```

(III.) COBOL

The third and final component in the banner creation process uses a COBOL program called 'MAKEBANN'. 'MAKEBANN' is responsible to create the banner page specific to the output spool file waiting to print and to construct the command file SI. SI is in turn used by SPOOK to append the banner page to the spool file.

THE BANNER PAGE

Below is a sample of the banner page without the mailing information. 'MAKEBANN' will be responsible to supply this information specific to each report.

```
               SEND REPORT TO:

               NAME:
               DEPT:
          MAIL CODE:


     FOR REPORT DISTRIBUTION UPDATES PLEASE CALL THE PROGRAMMING DEPT.
     OR UPDATE THIS PAGE AND MAIL TO THE PROGRAMMING DEPT.
```

Until now the output spool file MAILINFO was assumed to contain the mailing information for REPORT1. This assumption is no longer needed. Instead a file called 'USERINFO' will be used which contains the mailing information needed by the banner page. For this example 'USERINFO' is a flat MPE file, although it might well be a small data base or KSAM file. Regardless of the structure, the COBOL program will match the FNAME returned by the SHOWOUT command to a FNAME in 'USERINFO'. Here is where unique FNAMEs become important. For installations without unique FNAMEs the job stream of that particular job can be changed to equate either default FNAMEs (i.e. COBLIST, ASKLIST etc.) or conflicting FNAMEs to unique FNAMEs.

The layout and the information of the file 'USERINFO' is as follows:

```
FNAME    LSTNM                FSTNM        DEPT  MAILCD
REPORT1  RAINS                PAT          P81   204-36
REPORT3  PUBLIC               JOHN         W373  LOC-3A
REPORT4  HENRY                FRED         W454  LOC-21
```

'USERINFO' remains as a permanent disc file on the system and is updated periodically to include new jobs or to change existing information.

If the SHOWOUT command returns the following information:

```
DEV/CL   DFID    JOBNUM  FNAME    STATE FRM SPACE RANK PRI #C
137      #02219  #S580   REPORT1  READY        32    1  8   1
137      #02220  #S580   REPORT2  READY        32    2  8   1
137      #02221  #S580   REPORT3  READY        32    3  8   1
```

then 'MAKEBANN' will produce banner pages with mailing information for REPORT1 and REPORT3.

As an added feature the programmer might at this point want to send reports to multiple printers. With a litle extra work 'USERINFO' could contain several printer destinations. The COBOL program would check this field and make appropriate additions to the command file SI which is passed to SPOOK. For example if REPORT2 is to be printed at device 137 and at device 136 the command file SI would contain the following:

```
FILE BANNER;DEV=137,12,1
APPEND #DFID,#DFID;ALL,*BANNER
APPEND END
FILE BANNER;DEV=136,12,1
APPEND #DFID,#DFID;ALL,*BANNER
APPEND END
PURGE #DFID, #DFID
EXIT
```

THE PROCESS

The entire process runs as a background job. Once a minute the program wakes up to check the output spool files. These output spool files are compared to the FNAMEs of the 'USERINFO' file now stored in a table in 'MAKEBANN'. If a match occurs a banner page is generated with the mailing information. 'MAKEBANN' again checks the output spool files, this time to return the #DFID of the newly created banner page (BPAGE). The command file is then loaded and executed with the commands to append the banner page to the proper report. This process then repeats itself every minute.

SUMMARY

Reports can now be generated that facilitate distribution. The HP 3000 has readily available the tools to accomplish this task. The HP programmer no longer has to rely on the standard sysout page to route reports. Instead banner pages can now be created as they are needed for existing and future reports.

APPENDIX A
SOURCE LISTING FOR ALL PROGRAMS
PROGRAM: 'MAKEBANN'

```
$CONTROL NOLIST
IDENTIFICATION DIVISION.
PROGRAM-ID.  CREATE-BANNER.
*AUTHOR.  ECA.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. HP-3000.
OBJECT-COMPUTER. HP-3000.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
 SELECT 100-USERINFO-FILE         ASSIGN  'USERINFO'.
 SELECT 150-SPOOL-INFO-FILE       ASSIGN  'SO'.
 SELECT 200-SPOOK-COMMAND-FILE    ASSIGN  'SI'.
 SELECT 250-BANNER-PAGE-FILE      ASSIGN  'BPAGE'.
DATA DIVISION.
FILE SECTION.

FD  100-USERINFO-FILE
    RECORD CONTAINS 80 CHARACTERS.
01  100-USERINFO-RECORDS     PIC X(80).

FD  150-SPOOL-INFO-FILE
    LABEL RECORDS ARE STANDARD
    RECORD CONTAINS 80 CHARACTERS
    RECORDING MODE IS FIXED.
01  150-SPOOL-INFO-RECORD    PIC X(80).

FD  200-SPOOK-COMMAND-FILE
    LABEL RECORDS ARE STANDARD
    RECORD CONTAINS 80 CHARACTERS
    RECORDING MODE IS FIXED.
01  200-SPOOK-COMMAND-RECORD PIC X(80).

FD  250-BANNER-PAGE-FILE
    LABEL RECORDS ARE STANDARD
    RECORDS CONTAINS 133 CHARACTERS
    RECORDING MODE IS FIXED.
01  250-BANNER-PAGE-RECORD    PIC X(133).


WORKING-STORAGE SECTION.

01  200-COMMAND-LINE-1.
    05  FILLER              PIC X(80) VALUE SPACES.

01  200-COMMAND-LINE-2.
    05  FILLER              PIC X(25) VALUE.
        'FILE BANNER;DEV=137,13,'.
    05  200-COPIES          PIC X(04) VALUE SPACES.
```

```
01  200-COMMAND-LINE-3.
    05  FILLER              PIC X(07) VALUE 'APPEND '.
    05  200-DFID-BANNER     PIC X(06) VALUE SPACES.
    05  FILLER              PIC X(01) VALUE ','.
    05  200-DFID-REPORT     PIC X(06) VALUE SPACES.
    FILLER                  PIC X(12) VALUE
    ';ALL,*BANNER'.

01  200-COMMAND-LINE-4.
    05  FILLER              PIC X(10) VALUE 'APPEND END'.

01  200-COMMAND-LINE-5.
    05  FILLER                  PIC X(06) VALUE 'PURGE '.
    05  200-P-DFID-REPORT       PIC X(06) VALUE SPACES.
    05  FILLER                  PIC X(01) VALUE ','.
    05  200-P-DFID-BANNER       PIC X(06) VALUE SPACES.

01  250-BANNER-LINE-1.
    05  FILLER      PIC X(20) VALUE SPACES.
    05  FILLER      PIC X(16) VALUE 'SEND REPORT TO: '.
    05  FILLER      PIC X(97) VALUE SPACES.

01  250-BANNER-LINE-2.
    05  FILLER      PIC X(20) VALUE SPACES.
    05  FILLER      PIC X(06) VALUE 'NAME: '.
    05  250-B-LSTNM PIC X(20) VALUE SPACES.
    05  250-B-FSTNM PIC X(12) VALUE SPACES.
    05  FILLER      PIC X(75) VALUE SPACES.

01  250-BANNER-LINE-3.
    05  FILLER      PIC X(20) VALUE SPACES.
    05  FILLER      PIC X(06) VALUE 'DEPT: '.
    05  250-B-DEPT  PIC X(04) VALUE SPACES.
    05  FILLER      PIC X(103) VALUE SPACES.

01  250-BANNER-LINE-4.
    05  FILLER      PIC X(15) VALUE SPACES.
    05  FILLER      PIC X(11) VALUE 'MAIL CODE: '.
    05  250-B-DEPT  PIC X(06) VALUE SPACES.
    05  FILLER      PIC X(98) VALUE SPACES.

01  250-BANNER-LINE-5.
    05  FILLER      PIC X(10) VALUE SPACES.
    05  FILLER      PIC X(32) VALUE
    'FOR REPORT DISTRIBUTION UPDATES '.
    05  FILLER      PIC X(23) VALUE
    'PLEASE CALL PROGRAMMING'.
    05  FILLER      PIC X(68) VALUE SPACES.

01  250-BANNER-LINE-6.
    05  FILLER      PIC X(10) VALUE SPACES.
    05  FILLER      PIC X(29) VALUE
    'OR UPDATE THIS PAGE AND MAIL '.
    05  FILLER      PIC X(25) VALUE
    'TO THE PROGRAMMING DEPT.'.
    05  FILLER      PIC X(69) VALUE SPACES.
```

```
01  300-USERINFO-RECORD.
    05  300-FNAME            PIC X(08).
    05  FILLER               PIC X(01).
    05  300-LSTNM            PIC X(20).
    05  FILLER               PIC X(01).
    05  300-LSTNM            PIC X(12).
    05  FILLER               PIC X(01).
    05  300-DEPT             PIC X(04).
    05  FILLER               PIC X(01).
    05  300-MAILCD           PIC X(06).
    05  FILLER               PIC X(01).
    05  300-COPIES           PIC X(02).

01  310-SPOOK-SPOOL-RECORD.
    05  310-DEVICE           PIC X(08).
    05  FILLER               PIC X(01).
    05  310-DFID             PIC X(06).
    05  FILLER               PIC X(02).
    05  310-JOBNUM           PIC X(06).
    05  FILLER               PIC X(02).
    05  310-FNAME            PIC X(08).
    05  FILLER               PIC X(01).
    05  310-STATE            PIC X(06).
    05  FILLER               PIC X(06).
    05  310-SPACE            PIC X(06).
    05  FILLER               PIC X(04).
    05  310-RANK             PIC X(01).
    05  310-PRI              PIC X(02).
    05  310-COPIES           PIC X(04).

01  400-SWITCHES-AND-THINGS.
    05  USER-INFO-COUNT    PIC 9(02) VALUE 0.
    05  EOF-USER-INFO      PIC X(01) VALUE 'N'.
    05  EOF-SP             PIC X(01) VALUE 'N'.
    05  EOF-BPAGE          PIC X(01) VALUE 'N'.
    05  FIND-BPAGE         PIC X(01) VALUE 'N''
    05  FIND-USER          PIC X(01) VALUE 'N'.
    05  DFID-BANNER        PIC X(06) VALUE SPACES.
    05  DFID-REPORT        PIC X(06) VALUE SPACES.
    05  COPIES             PIC X(04) VALUE SPACES.
    05  TABLE-SEARCH       PIC X(01) VALUE 'N'.
    05  END-OF-TIME        PIC X(01) VALUE 'N'.

01  500-USER-INFO-TABLE.
    05  500-USER-INFO OCCURS 1 TO 100 TIMES
                      DEPENDING ON USER-INFO-COUNT
                      ASCENDING KEY IS 500-FNAME
                      INDEXED BY USER-IDX.
    10  500-FNAME                PIC X(08).
    10  FILLER                   PIC X(01).
    10  500-LSTNM                PIC X(20).
    10  FILLER                   PIC X(01).
    10  500-FSTNM                PIC X(12).
    10  FILLER                   PIC X(01).
    10  500-DEPT                    PIC X(04).
    10  FILLER                      PIC X(01).
    10  500-MAILCD                  PIC X(06).
    10  FILLER                      PIC X(01).
    10  500 COPIES                  PIC X(02).


PROCEDURE DIVISION.

A000-MAIN.
     OPEN INPUT 100-USERINFO-FILE.
     SET USER-IDX TO 1.
     READ 100-USERINFO-FILE INTO 300-USERINFO-RECORD
        AT END MOVE 'Y' TO EOF-USER-INFO.
     PERFORM A020-LOAD-USER-INFO-TABLE UNTIL
                          EOF-USER-INFO = 'Y'.
     CLOSE 100-USERINFO-FILE.
     PERFORM A035 PROCESS UNTIL END-OF-TIME = 'Y'.

A035-PROCESS.
     CALL 'WAIT'.
     PERFORM A030-LOAD-SHOWOUT-INFO.
     CALL 'CALLSPK'.

     MOVE 'N' TO TABLE-SEARCH.
     MOVE 'N' TO FIND-USER.
     MOVE 'N' TO EOF-SP.
     SET USER-IDX TO 1.
     OPEN I-O   150-SPOOL-INFO-FILE.
     PERFORM A040-READ-SPOOL-INFO-FILE
           UNTIL EOF-SP = 'Y' OR FIND-USER = 'Y'.
     CLOSE 150-SPOOL-INFO-FILE.
     IF FIND-USER = 'Y'
     THEN
        PERFORM A050-CREATE-BANNER-PAGE
     ELSE
        GO TO A035-PROCESS.

     CALL 'CALLSPK'.

     MOVE 'N' TO EOF-BPAGE.
     MOVE 'N' TO FIND-BPAGE.
     OPEN I-O 150-SPOOL-INFO-FILE.
     READ 150-SPOOL-INFO-FILE INTO
        310-SPOOK-SPOOL-RECORD AT END
         MOVE 'Y' TO EOF-BPAGE.
     PERFORM A060-FIND-DFID-BPAGE UNTIL
       EOF-BPAGE = 'Y' OR FIND-BPAGE = 'Y'.
     CLOSE 150-SPOOL-INFO-FILE.

     PERFORM A070-PREPARE-APPEND-COMMANDS.

     CALL 'CALLSPK'.

     GO TO A035-PROCESS.
```

```
A020-LOAD-USER-INFO-TABLE.
     ADD 1 TO USER-INFO-COUNT.
     MOVE 300-FNAME  TO 500-FNAME (USER-INFO-COUNT).
     MOVE 300-LSTNM  TO 500-LSTNM (USER-INFO-COUNT).
     MOVE 300-FSTNM  TO 500-FSTNM (USER-INFO-COUNT).
     MOVE 300-DEPT   TO 500-DEPT (USER-INFO-COUNT).
     MOVE 300-MAILCD TO 500-MAILCD (USER-INFO-COUNT).
     MOVE 300-COPIES TO 500-COPIES (USER-INFO-COUNT).

     SET USER-IDX UP BY 1.
     READ 100-USERINFO-FILE INTO 300-USERINFO-RECORD
        AT END MOVE 'Y' TO EOF-USER-INFO.

A030-LOAD-SHOWOUT-INFO.
     OPEN I-O 200-SPOOK-COMMAND-FILE.
     MOVE 'SHOWOUT JOB=@;DEV=LP226' TO
          200-COMMAND-LINE-1.
     WRITE 200-SPOOK-COMMAND-RECORD FROM
           200-COMMAND-LINE-1.
     MOVE 'EXIT' TO 200-COMMAND-LINE-1.
     WRITE 200-SPOOK-COMMAND-RECORD FROM
           200-COMMAND-LINE-1.
     CLOSE 200-SPOOK-COMMAND-FILE.

A040-READ-SPOOL-INFO-FILE.
     READ 150-SPOOL-INFO-FILE INTO
          310-SPOOL-RECORD AT END
          MOVE 'Y' TO EOF-SP.
     IF 310-FNAME = 'STOPSPK'
     THEN
        PERFORM A080-END.
     IF 310-STATE = 'READY'
     THEN
      SEARCH ALL 500-USER-INFO
       AT END MOVE 'Y' TO TABLE-SEARCH
        WHEN 500-FNAME (USER-IDX) = 310-FNAME
         MOVE 310-DFID TO DFID-REPORT
         MOVE 500-LSTNM (USER-IDX)  TO 250-B-LSTNM
         MOVE 500-FSTNM (USER-IDX)  TO 250-B-FSTNM
         MOVE 500-DEPT (USER-IDX)   TO 250-B-DEPT
         MOVE 500-MAILCD (USER-IDX) TO 250-B-CODE
         MOVE 310-COPIES TO COPIES
         MOVE 'Y' TO FIND-USER
         MOVE 'Y' TO EOF-SP.

A050-CREATE-BANNER-PAGE.
     OPEN OUTPUT 250-BANNER-PAGE-FILE
     WRITE 250-BANNER-PAGE-RECORD FROM
           250-BANNER-LINE-1 AFTER 10.
     WRITE 250-BANNER-PAGE-RECORD FROM
           250-BANNER-LINE-2 AFTER 2.
     WRITE 250-BANNER-PAGE-RECORD FROM
           250-BANNER-LINE-3.
     WRITE 250-BANNER-PAGE-RECORD FROM
     WRITE 250-BANNER-PAGE-RECORD FROM
           250-BANNER-LINE-4.
     WRITE 250-BANNER-PAGE-RECORD FROM
           250-BANNER-LINE-5 AFTER 5.
     WRITE 250-BANNER-PAGE-RECORD FROM
           250-BANNER-LINE-6.
     CLOSE 250-BANNER-PAGE-FILE.

A060-FIND-DFID-BPAGE.
     IF 310-FNAME = 'BPAGE'
     THEN
        MOVE 310-DFID TO DFID-BANNER
         MOVE 'Y' TO FIND-BPAGE.

      IF FIND-BPAGE = 'N'
      THEN
      READ 150-SPOOL-INFO-FILE INTO
           310-SPOOK-SPOOL-RECORD AT END
            MOVE 'Y' TO EOF-BPAGE.

A070-PREPARE-APPEND-COMMANDS.
     OPEN I-O 200-SPOOK-COMMAND-FILE.
     MOVE COPIES      TO 200-COPIES
     MOVE DFID-BANNER TO 200-DFID-BANNER,
                         200-P-DFID-BANNER.
     MOVE DFID-REPORT TO 200-DFID-REPORT,
                         200-P-DFID-REPORT.
     WRITE 200-SPOOK-COMMAND-RECORD FROM
           200-COMMAND-LINE 2.
     WRITE 200-SPOOK-COMMAND-RECORD FROM
           200-COMMAND-LINE 3.
     MOVE SPACES TO 200-COMMAND-LINE-1.
     WRITE 200-SPOOK-COMMAND-RECORD FROM
           200-COMMAND-LINE 5.
     MOVE 'EXIT' TO 200-COMMAND-LINE-1.
     WRITE 200-SPOOK-COMMAND-RECORD FROM
           200-COMMAND-LINE-1.
     CLOSE 200-SPOOK-COMMAND-FILE.

A080-END.
     CLOSE 150-SPOOL-INFO-FILE.
     CLOSE 200-SPOOK-COMMAND-FILE.
     CLOSE-250-BANNER-PAGE-FILE.
     GOBACK.
```

PROGRAM: 'CALLSPK'

```
$CONTROL SUBPROGRAM
BEGIN
PROCEDURE CALLSPK;
BEGIN
INTRINSIC CREATEPROCESS, ACTIVATE, KILL;
OWN ARRAY B (0:2);
OWN INTEGER ARRAY IA (0:2) :=1(9,8,0);
OWN LOGICAL ARRAY LA (0:1);
OWN BYTE ARRAY BA1 (0:39) := 1('SO.PUB.PRODW',%15);
OWN BYTE ARRAY BA2 (0:39) := 1('SI.PUB.PRODW',%15);
OWN BYTE ARRAY PRINTPROC (0:39):=
   'SPOOK5.PUB.SYS';
INTEGER ERROR,PIN;
LA(0) :=@BA1;
LA(1) :=@BA2;
CREATEPROCESS (ERROR,PIN,PRINTPROC,IA,LA);
ACTIVATE (PIN,2);
KILL (PIN);
END;
END.
```

PROGRAM: 'WAIT'

```
$CONTROL SUBPROGRAM
BEGIN
INTRINSIC PAUSE;
PROCEDURE WAIT;
BEGIN
REAL A:=6.0E+1;
PAUSE (A);
END;
END.
```

APPENDIX B
SOURCE LISTING FOR ALL JOB STREAMS
JOB STREAM : 'STARTUP'

```
! JOB STARTUP
! OUTFENCE 12;LDEV=137
! HEADOFF 137
! PURGE SO
! PURGE SI
! BUILD SO;REC=-80,,F,ASCII;DISC=100
! BUILD SI;REC=-80,,F,ASCII;DISC=10
! FILE BPAGE;REC=-133,F,ASCII;DEV=137,1,1
! RUN PROCESS1
! EOJ
```

To create the run module 'PROCESS1' all the programs must be compiled into the same USL (User Segmented Library) as follows:

```
:COBOLII MAKEBANN, PROCESS
:SPL CALLSPK, PROCESS
:SPL WAIT, PROCESS
```

Next the USL file 'PROCESS' is prepared with PH (Process Handling) capability.

```
:PREP PROCESS, PROCESS1;CAP=PH
:SAVE PROCESS1
```

JOB STREAM : 'SHUTDOWN'

```
! JOB SHUTDOWN
! FILE STOPSPK;DEV=137,1,1
! LISTF any file;*STOPSPK
! OUTFENCE 1;LDEV=137
! TELLOP HEADON 137
! EOJ
```

TITLE: How To Be Sucessful in Your Implementation of Any Office Technology

AUTHOR: Susan Wyatt

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 0186

Bar Codes in Factory Data Collection

Terry W. Simpkins
Spectra-Physics
Retail Systems Division
959 Terry Street
Eugene, OR 97402

Over the last couple of years, much has been written about the impending bar code revolution in U.S. business. While bar codes are being used extensively in retail applications (primarily grocery stores), they have been relatively slow to catch on in manufacturing circles. There are several reasons why acceptance has been slow, including a perceived difficulty in implementation. In Hewlett-Packard 3000 shops, difficulty is not a legitimate excuse anymore. This paper describes one implementation of factory bar-code use, the goals of the project, and the methods used to meet those goals.

Our primary goal was to capture data throughout the assembly process of a new product. The data we wished to capture included: unit serial number, several component part serial numbers, results of various tests, and actions taken to resolve problems and test failures. The constraints placed on the process included: 100% accuracy in data entry, quick data entry, and response time that would not impede the assembly process with volumes up to 300 units per day. Once test data was collected on a significant population of units, we would generate SPC charts on the various production processes.

The general layout of the production process is outlined in figure #1. As you can see, it is a linear process with a vary well-defined workflow. This is a major contributor to the simplicity of the process. The assembly method required data entry at nine positions along the production line and data inquiry was needed at several other locations.

Two of the data entry stations presented a special challenge because they are dedicated PCs performing automated testing on the units. These PCs made physical connections to the units, performed functional and specification compliance tests, graded performance, passed or rejected the unit, and recorded the data collected and decisions made. If the units are rejected the data collected is examined at the network station to facilitate troubleshooting and determine appropriate repair of the units. This means a multi-user/multi-tasking environment is required. The software on the PC required to perform these tests would be written in-house.

Early on in the needs definition, it was decided to continue using an existing PC-based package to generate the required SPC graphs. This meant we needed the ability to extract data from whatever system was used to collect and store the data and transfer it to a PC via ASCII file. Four alternatives were identified and evaluated:

1) Use the shop-floor control and quality modules of our integrated manufacturing package. This did not provide the functionality or response time guarantee required. Additionally, it was determined the coding required to interface with the dedicated PC test stations would be too difficult to maintain through multiple releases of the vendor's package. Also, this option would require the new software to run on our current Series 70; however, current workload on the 70 would not allow us to guarantee the required response time.

2) Develop a PC/LAN-based system designed specifically to accommodate our needs. This alternative was rejected due to the lack of in-house MS-DOS expertise and the lack of a true multi-tasking, multi-user database solution for the PCs.

3) Purchase a third-party solution. No existing solution was identified that would meet our specifications without substantial modification. Additionally, all packages we found were written in a language we didn't know, or they were priced at an unacceptably high level (i.e., over $30,000).

4) Create our own "home-grown" system designed to provide the needed functionality, and no more. This is the approach we adopted.

We chose to base our system on the HP3000 since that is where our expertise base. The assumption being slightly more expensive hardware and software would be offset by the reduction in training and development time, as well as the cost of on-going support.

We next decided on the Powerhouse product being our language since it is the fourth GL we own and use. These two decisions allowed us to begin development with no delay. The system requirements were broken down into several modules:

1. Basic data collection
   - Part Numbers and Serial Numbers
   - Manual Tests
   - Rework Activity
2. Automated Tests
3. Data Inquiry/Look Ups
4. Label Printing

By using Powerhouse, we were able to create prototype screens for data collection, manual tests, and rework very quickly. These prototypes allowed us to fine-tune our design as we went and thereby avoid significant "mid-course" corrections and massive rewrites.

The data communications requirements of talking to the PCs were more complex. We used COBOL for the programs that would accept data from the PCs and update the host database.

This allowed easy access to all intrinsics and again required very little learning time (for an in-depth discussion on programmatic terminal access, see the "Data Comm" column in the February and April 1985 issues of Interact).

For creation of bar code menus we chose the HP laserjet and the ASK Bar-Scan product. Bar-Scan allows easy creation of bar coded menus and documents. Initially, we used Quiz to create bar-codes on an HP2563 printer for testing. The decision to switch to a stock piece of third-party software was made to reduce development and support requirements. Conversion to a laserjet improved appearance, speed, and reduced the cost. The font cartridge required to create code 3 of 9 bar codes on the laserjet is HP92286W1, commonly referred to as W1.

To read the bar codes we selected the Intermec model 9510 series of "online readers." This model was chosen for several reasons. The Intermec 9510 supports light wand or laser scanners, and can be easily programmed to add carriage returns or enter keys after the number is read. This makes it much easier to use standard bar codes in existing applications and eliminate the manual use of keyboards. The 9510 is modular in design, you may replace the wand reader, the cord, the decoder, or the power supply individually instead of the entire unit. Most importantly, the Intermec unit goes between the host and the terminal using standard 25-pin connectors which allows the same unit to be used with several different terminal types.

Because the host can't tell where the data came from (wand or keyboard), no special programming efforts or techniques are required, except for configuring the reader. The Powerhouse screens developed for data collection are the same if we use keyboards for manual entry or wands to read bar codes.

To overcome our response time constraint we decided to give this system its own dedicated HP3000. While this seems extreme, the cost was actually very competitive to a networked PC approach. Figure 2 lists the hardware configuration and costs.

Because this system is virtually stand-alone, we determined an INP link to our current Series 70 would easily satisfy our communication needs. This link will eventually be converted to LAN/3000. The MICROXE platform was chosen because of its expandability and its longer expected life. The standard MICRO being recently replaced by the LX and GX models.

To print unit serial number labels, we selected the HP2934. Because of its low cost, high quality bar codes and the outstanding reliability of the unit, it was an obvious choice given our desire to keep configurations as simple as possible.

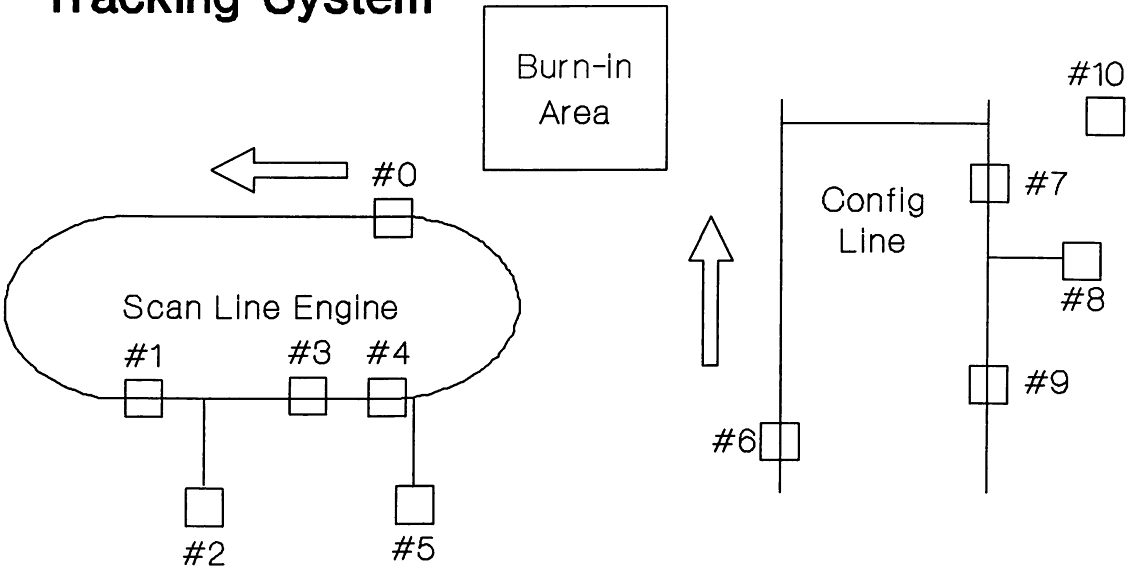
As you can see, bar coding applications are easy! Scenarios are possible that require no software changes to the data input function and several methods exist for creating bar coded output with minimal effort and expense. Our system consumed less than 2 man-months from design to turn on. This relatively short time is directly attributable to the use of a 4th generative language for program development and the Intermec readers being transparent to the HP3000.

To help you get started and understand how to create your own bar-coding system, Figure 3 contains a schematic of the database. Figure 4 shows the layout and source code for two of the screens used to collect data.

The advantages to be gained by using bar codes can be significant if absolute data accuracy is important in your applications, so get on the wagon now!

Freedom Assembly Tracking System

Figure 1


Burn-in Area
#10
Config Line
#7
#8
#9
#6
#0
Scan Line Engine
#1
#3
#4
#2
#5

* #0 Start a Unit
* #1 Beamwalk Test
* #2 Beamwalk Rework
* #3 Motor & Main PCB Installation
#4 Scan Engine Test (2 PCs)
* #5 Scan Engine Rework

* #6 Configuration
#7 Final Test
* #8 Final Test Rework
#9 Photon Dr-Test
* #10 Shipping
* #11 Service (not shown)

* = Terminal/wand

Hardware Configuration List

Figure 2

HP PART #	DESCRIPTION	List Price	Qty	Ext Price
32545A	MICRO XE w/ 4MB & PIC	$30,500	1	$30,500
40290A opt 125	ATP/M 4-25 PIN MODEM & 4-RS232 DIRECT CONNECT	$3,780	2	$7,560
7958A	130MB DISC DRIVE	$6,450	1	$6,450
C1001G	HP 700/92 TERMINAL	$895	9	$8,055
9144A	1/4" CARTRIDGE TAPE	$2,350	1	$2,350
30284A	NS Pt-to-Pt 3000/V LINK	$0	1	
opt 110	SYNC MODEM	$3,085	1	$3,085
32344A	NS3000/V NETWORK SERVICES	0	1	$0
opt 310	for MICRO XE	2040	1	$2,040
2934A	HP Printer (200cps)	2995	1	$2,995
	INTERMEC Model 9510	660	9	$5,940
	Wand Readers (complete)			$0
		Total =		$68,975

This cost can be drastically reduced through one of several means:

1) Purchase used equipment in the after market or from HP's Remarketed Division
2) Use a VAR who will give you a volume discount
3) Purchase HP DEMO equipment.
4) Utilize current equipment where possible

Note: If we hadn't needed a dedicated processor, our total cost would have been: $16,990.

Figure 3
Page 1

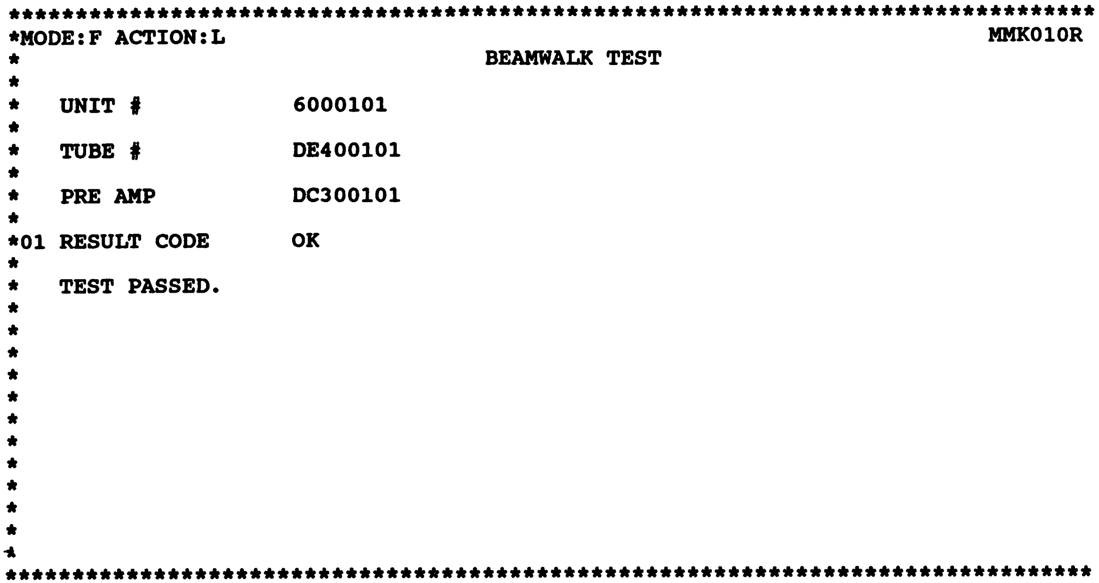
```
DATA BASE: FREE  .JOE


DATA BASE LANGUAGE ATTRIBUTE: NATIVE-3000


SET NAME:
   M-FAILURE,MANUAL

      ITEMS:
         FAIL-CODE,             X4             <<KEY ITEM>>
         DESCRIPTION,           X72

      CAPACITY: 251              ENTRIES: 20


SET NAME:
   M-REWORK,MANUAL

      ITEMS:
         REWORK-CODE,           X4             <<KEY ITEM>>
         DESCRIPTION,           X72

      CAPACITY: 101              ENTRIES: 4


SET NAME:
   M-SERIAL,MANUAL

      ITEMS:
         SERIAL-NO,             X8             <<KEY ITEM>>

      CAPACITY: 5003             ENTRIES: 271


SET NAME:
   M-TARGET,MANUAL

      ITEMS:
         PROD-DATE,             X6             <<KEY ITEM>>
         FIRST-UNIT,            X8
         LAST-UNIT,             X8
         TARGET-QTY,            J1
         ACTUAL-QTY,            J1

      CAPACITY: 503              ENTRIES: 10


SET NAME:
   M-UNIT,MANUAL

      ITEMS:
         UNIT,                  X8             <<KEY ITEM>>
         TUBE,                  X8
         PRE-AMP,               X8
         MOTOR,                 X8
         MAIN-PCB,              X8
         DECODER,               X8
         MODEL,                 X6
         POWER-BRICK,           X10
         CABLE,                 X10
         SOFTWARE,              X10
         T1-FLAG,               X2
         T2-FLAG,               X2
         T3-FLAG,               X2
         TIME-0,                X12
         TIME-1,                X12
         TIME-3,                X12
         TIME-4,                X12
         TIME-6,                X12
         TIME-7,                X12

      CAPACITY: 1009             ENTRIES: 85


SET NAME:
   M-XREF,MANUAL

      ITEMS:
         XREF,                  X6             <<KEY ITEM>>

      CAPACITY: 101              ENTRIES: 11


SET NAME:
   D-FAILURE,DETAIL

      ITEMS:
         FAIL-CODE,             X4             <<SEARCH ITEM>>
         SEQ,                   X2             <<SORT ITEM>>
         F-LINE,                X72

      CAPACITY: 2021             ENTRIES: 5


SET NAME:
   D-TEST-1,DETAIL

      ITEMS:
         UNIT,                  X8             <<SEARCH ITEM>>
         TIME-T1,               X12            <<SORT ITEM>>
         FAIL-CODE-T1,          X4
         REWORK-T1,             X4
         TUBE,                  X8
         PRE-AMP,               X8

      CAPACITY: 3042             ENTRIES: 82


SET NAME:
   D-TEST-2,DETAIL

      ITEMS:
         UNIT,                  X8             <<SEARCH ITEM>>
         TIME-T2,               X12            <<SORT ITEM>>
         FAIL-CODE-T2,          X4
         REWORK-T2,             X4
         TUBE,                  X8
         PRE-AMP,               X8
         MOTOR,                 X8
         MAIN-PCB,                X8
         TD-POWER-2-FLAG,         X2
         TD-POWER-2,              J1
         LINE-SEP-FLAG,           X2
         LINE-SEP,               4J1
         FWHM-FLAG,               X2
         FWHM-FF,               20J1
         FWHM-W,                20J1
         FWHM-L,                  J1
         CENTROID-FLAG,           X2
         CENTROID-FF,           20J1
         CENTROID-W,            20J1
         GLITCH-FLAG,             X2
         GLITCH,                20J1
         FIRST-DRV-FLAG,          X2
         FIRST-DRV,             20J1

      CAPACITY: 3000              ENTRIES: 65


SET NAME:
   D-TEST-3,DETAIL

      ITEMS:
         UNIT,                    X8             <<SEARCH ITEM>>
         TIME-T3,                 X12            <<SORT ITEM>>
         FAIL-CODE-T3,            X4
         REWORK-T3,               X4
         DECODER,                 X8
         POWER-UP-FLAG,           X2
         AC-IN,                   J1
         DC-OUT,                  J1
         RIPPLE,                  J1
         TD-POWER-3-FLAG,         X2
         TD-POWER-3,              J1
         MO-SPEED-FLAG,           X2
         MO-SPEED,                J1
         READ-A-FLAG,             X2
         READ-E-FLAG,             X2
         PORT-FLAG,               X2
         INTERFACE-FLAG,          X2
         BEEP-FLAG,               X2
         LED-FLAG,                X2
         EEPROM-FLAG,             X2
         EPROM-VERSION,           X12

      CAPACITY: 3008              ENTRIES: 1
```

PATH IDENTIFYING INFORMATION

MASTER SET NAME	ASSOCIATED DETAIL SET NAME	SEARCH ITEM NAME	SORT ITEM NAME
M-FAILURE	D-FAILURE	FAIL-CODE	SEQ
M-REWORK			
M-SERIAL			
M-TARGET			
M-UNIT	D-TEST-1 D-TEST-2 D-TEST-3	UNIT UNIT UNIT	TIME-T1 TIME-T2 TIME-T3
M-XREF			

DETAIL SET NAME	SEARCH ITEM NAME	SORT ITEM NAME	ASSOCIATED MASTER SET NAME
D-FAILURE	!FAIL-CODE	SEQ	M-FAILURE
D-TEST-1	!UNIT	TIME-T1	M-UNIT
D-TEST-2	!UNIT	TIME-T2	M-UNIT
D-TEST-3	!UNIT	TIME-T3	M-UNIT

Figure 4
Page 1



```
********************************************************************************
*MODE:F ACTION:L                                                          MMK010R
*                                   BEAMWALK TEST
*
*   UNIT #              6000101
*
*   TUBE #              DE400101
*
*   PRE AMP             DC300101
*
*01 RESULT CODE         OK
*
*   TEST PASSED.
*
*
*
*
*
*
*
*
*
*
*
*
********************************************************************************
```

```
********************************************************************************
*MODE:F ACTION:L                                                        MMK060R
*                                  CONFIGURATION
*
*   UNIT #
*
*   MODEL #
*
*   DECODER NUMBER
*
*   POWER BRICK
*
*   CABLE NUMBER
*
*   SOFTWARE
*
*
*
*
*
*
*
*
*
*OPERATION INCONSISTENT WITH ACCESS TYPE  (FSERR 40) (BUFFER*01)
********************************************************************************
```

Figure 4
Page 2

```
SCREEN MMK010R
FILE D-TEST-1 ALIAS TEST-1 NODELETE
FILE M-UNIT SECONDARY NOITEMS NODELETE
     ACCESS VIA UNIT USING UNIT OF TEST-1
FILE M-SERIAL DESIGNER ALIAS SERIAL-T
FILE M-SERIAL DESIGNER ALIAS SERIAL-P
FILE D-TEST-1 DESIGNER ALIAS TEST-X
FILE M-FAILURE REFERENCE
     ACCESS USING FAIL-CODE-T1
FILE M-XREF REFERENCE
ITEM TIME-T1 FINAL ASCII(SYSDATE,6) + ASCII(SYSTIME / 100, 6)
ITEM T1-FLAG &
     FINAL "P1" IF FAIL-CODE-T1 ="OK" AND T1-FLAG = "   " &
      ELSE "P " IF FAIL-CODE-T1 ="OK" AND T1-FLAG <> "   " &
      ELSE "F "
ITEM TUBE OF M-UNIT FINAL TUBE OF TEST-1
ITEM PRE-AMP OF M-UNIT FINAL PRE-AMP OF TEST-1
ITEM TIME-1 FINAL ASCII(SYSDATE,6) + ASCII(SYSTIME / 100, 6) &
     IF TIME-1 = "            "
HILITE DATA INVERSE, DISPLAY INVERSE HALFTONE, &
       MESSAGE INVERSE, ERROR INVERSE BLINKING AUDIBLE
TITLE "MMK010R" AT 1,72
TITLE "BEAMWALK TEST" AT 2,41 CENTERED
SKIP 1
FIELD UNIT OF TEST-1 NOID SIZE 7 REQUIRED &
      LOOKUP ON M-UNIT MESSAGE 10
SKIP 1
FIELD TUBE OF TEST-1 NOID REQUIRED &
      LOOKUP ON M-XREF USING "TU" + TUBE [1:2] MESSAGE 11
SKIP 1
FIELD PRE-AMP OF TEST-1 NOID REQUIRED &
      LOOKUP ON M-XREF USING "PA" + PRE-AMP [1:2] MESSAGE 12
SKIP 1
FIELD FAIL-CODE-T1 LABEL "RESULT CODE" REQUIRED &
      LOOKUP ON M-FAILURE MESSAGE 13
SKIP 1
FIELD DESCRIPTION NOID NOLABEL DATA AT ,4 DISPLAY
PROCEDURE PROCESS UNIT
  BEGIN
    WHILE RETRIEVING TEST-X BACKWARDS USING UNIT OF TEST-1
      BEGIN
        IF FAIL-CODE-T1 OF TEST-X <> "OK" AND REWORK-T1 OF TEST-X = "
          THEN BEGIN
            ERROR 14
            BREAK
            END
        END
    END
PROCEDURE EDIT TUBE
  BEGIN
    IF FIELDTEXT <> TUBE OF M-UNIT
      THEN BEGIN
        GET SERIAL-T USING FIELDTEXT OPTIONAL
        IF ACCESSOK
          THEN ERROR 15
        END
    END
```

Figure 4
Page 4

```
PROCEDURE EDIT PRE-AMP
  BEGIN
    IF FIELDTEXT <> PRE-AMP OF M-UNIT
      THEN BEGIN
        GET SERIAL-P USING FIELDTEXT OPTIONAL
        IF ACCESSOK
          THEN ERROR 16
        END
    END
PROCEDURE PREUPDATE
  BEGIN
    IF TUBE OF M-UNIT <> "        "
      THEN GET SERIAL-T USING TUBE OF M-UNIT
    LET SERIAL-NO OF SERIAL-T = TUBE OF TEST-1
    IF PRE-AMP OF M-UNIT <> "        "
      THEN GET SERIAL-P USING PRE-AMP OF M-UNIT
    LET SERIAL-NO OF SERIAL-P = PRE-AMP OF TEST-1
    END
PROCEDURE UPDATE
  BEGIN
    STARTLOG TEST-1
    PUT NOTDELETED M-UNIT
    PUT TEST-1
    PUT M-UNIT
    PUT SERIAL-P
    PUT SERIAL-T
    STOPLOG
    END
PROCEDURE DESIGNER 01
  BEGIN
    ACCEPT FAIL-CODE-T1
    DISPLAY DESCRIPTION
    END
```

Figure 4
Page 5

```
SCREEN MMK060R AUTOUPDATE
FILE BUFFER NODELETE OPEN WRITE EXCLUSIVE
     TARGET 0
FILE M-UNIT SECONDARY NOITEMS NODELETE
     ACCESS VIA UNIT USING UNIT OF BUFFER
FILE D-TEST-2 DESIGNER
FILE M-SERIAL DESIGNER
FILE M-XREF REFERENCE
TEMPORARY FIRST-FLAG CHAR*1 INITIAL " " RESET AT STARTUP
ITEM MODEL OF M-UNIT FINAL MODEL OF BUFFER
ITEM DECODER OF M-UNIT FINAL DECODER OF BUFFER
ITEM TIME-6 OF M-UNIT &
     FINAL ASCII(SYSDATE,6) + ASCII(SYSTIME / 100, 6) &
           IF TIME-6 OF M-UNIT = "      "
HILITE DATA INVERSE, DISPLAY INVERSE HALFTONE, &
       MESSAGE INVERSE, ERROR INVERSE BLINKING AUDIBLE
TITLE "MMK060R" AT 1,72
TITLE "CONFIGURATION" AT 2,41 CENTERED
SKIP 1
ALIGN (,4,21)
FIELD UNIT OF BUFFER SIZE 7 REQUIRED LOOKUP ON M-UNIT MESSAGE 10
SKIP 1
FIELD MODEL OF BUFFER SIZE 5 REQUIRED
SKIP 1
FIELD DECODER OF BUFFER REQUIRED &
      LOOKUP ON M-XREF USING "DC" + DECODER [1:2] MESSAGE 61, &
        ON M-XREF USING "DC" + DECODER [1:2] + MODEL [1:2] MESSAGE 63
SKIP 1
FIELD POWER-BRICK OF BUFFER LABEL "POWER BRICK" REQUIRED
SKIP 1
FIELD CABLE OF BUFFER REQUIRED
SKIP 1
FIELD SOFTWARE OF BUFFER REQUIRED
PROCEDURE PROCESS UNIT
  BEGIN
    LET FIRST-FLAG = "Y"
    WHILE RETRIEVING D-TEST-2 BACKWARDS USING UNIT OF BUFFER
      BEGIN
        IF FAIL-CODE-T2 <> "OK"
          THEN BEGIN
            IF FIRST-FLAG = "Y"
              THEN BEGIN
                ERROR 62
                BREAK
                END
            ELSE BEGIN
              IF REWORK-T2 = "    "
                THEN BEGIN
                  ERROR 65
                  BREAK
                  END
              END
            END
        LET FIRST-FLAG = "N"
        END
    END
PROCEDURE PROCESS MODEL
```

```
  BEGIN
    IF NOT MATCHPATTERN(TRUNCATE(MODEL),"6(6|7|8|9)(0|1)(0|1|2)(0|2)")
      THEN ERROR 64
    END
PROCEDURE PROCESS DECODER
  BEGIN
    IF DECODER OF M-UNIT <> DECODER OF BUFFER
      THEN BEGIN
        GET M-SERIAL USING DECODER OF BUFFER OPTIONAL
        IF ACCESSOK
          THEN ERROR 60
        END
    END
PROCEDURE PREUPDATE
  BEGIN
    IF DECODER OF M-UNIT <> "        "
      THEN GET M-SERIAL USING DECODER OF M-UNIT
    LET SERIAL-NO OF M-SERIAL = DECODER OF BUFFER
    END
PROCEDURE UPDATE
  BEGIN
    STARTLOG M-UNIT
    PUT NOTDELETED M-UNIT
    PUT M-UNIT
    PUT M-SERIAL
    STOPLOG
    END
```

A Development Methodology for a New Generation

by Grant W. Fletcher of The Interface Group, Incorporated, and
Kathleen A. Sachara of The Haley Corporation

Abstract of the Paper

The traditional methodology applied to system development has been eclipsed by the advanced software technology applied in the current generation of computer programming languages, thereby creating the need for these standards to be re-evaluated with regard to non-procedural programming languages and re-defined, where necessary, to provide an environment in which productive development and recognized standards for maintenance and auditability may co-exist.

A Development Methodology for a New Generation is an evaluation of the traditional development approach against the reality of programming in a non-procedural language. The observations and conclusions of this exercise form the basis of what is later presented as one proposed methodology in which the benefits of development in a non-procedural language and standards for maintenance and auditability are recognized as equally critical factors of successful system development.

The objective of the paper is to raise general awareness of the strengths and weaknesses of traditional development methodology with regard to current software development tools and languages.

Agenda of the Presentation

Traditional Methodology for System Development.

A review of the traditional standards for system development and the traditional project life cycle is presented, giving the audience a point of common reference for the discussion that follows.

Traditional Methodology Applied to a Non-Traditional Development.

A case study is presented in which a project is governed by traditional system development methodology and standards, but which involves code developed in a non-procedural language and the use of other software development tools.

Observations and Conclusions of the Exercise.

The strengths and weaknesses of the traditional approach to system development applied in a non-procedural language environment are discussed with reference to the preceding case study.

A Proposed Methodology.

The strengths of the traditional approach are augmented with revisions and additions to deal with the weaknesses of this approach found in the preceding case study. Specifically, the auditability and maintainability of the traditional methodology is enhanced with proposed standards for prototyping and code generation.

Traditional Methodology for System Development.

Development of computer based systems has evolved to a certain level of maturity with which potential users and data processing auditors are somewhat comfortable, but with the tools and techniques of such many data processing professionals find their creativity and productivity suppressed and frustrated. Traditional methodologies provide the user with reasonable certainties by which they may plan future staffing requirements and functions, and defineable project life cycles by which auditors may measure development responsiveness. A project life cycle built from a traditional system development approach provides comfort for the business manager and data processing auditor involved with a system implementation. The following is an example of one such project life cycle.

A Traditional Project Life Cycle

1. Project Initiation

6. Implementation   2. Systems Analysis

5. Development   3. Project Proposal

4. Project Specification

The first feature to notice about any traditional project life cycle is that each task is a defined step in the process of completing the project, and that all tasks, except the first, is contigent upon the completion of it's predecessor. The second important aspect of any traditional approach to system development is that project initiation originates from a user request for some function to be computerized, for some existing computer system to be enhanced, or for some existing computer system to be modified to resolve an error or problem.

1. Project Initiation. The catalyst for all system development activity under a traditional methodology is some form of request from a user, or potential user. A comprehensive methodology would require that the request be documented in some formal manner and logged to provide an audit trail. The user would be expected to provide a large amount of detail about their request. This may include a description of the function involved, how and why it is currently performed, who performs the procedures of the function, and what benefit they may expect from the implementation of the requested work. One may expect that this request would originate at a very low level on the corporate hierarchy and have to work it's way through several levels of supervisory personnel and management before it is actually submitted to systems development for consideration.

2. Systems Analysis. Since the project is initiated by a user, who should not be expected to fully understand the implications of their request throughout the organization as a whole, and who may not be aware of all available alternatives that may address the requirement that they have raised, a member of the systems development team will prepare an analysis of the request, as presented. A thorough systems analysis should entail a complete

description of the function involved and it's relationship to other elements of the organization, a compilation of all available alternatives with limited judgement on their individual merits and demerits, an analysis of the information required and generated by the function, an analysis of the procedures required or currently used to perform the function, a description of the hardware used or required to automate the task, and an evaluation of the future evolution and maintenance of any implemented system.

3. Project Proposal. The systems analysis produced above is used to form the basis of a project proposal which isolates the alternative from the systems analysis that best suits the requirements of the user and conforms to all relevant company policies and other procedures. The project proposal is written in language that is understandable by the users concerned with the implementation of the project, and describes information, procedures, process environment, implementation, and methodology for future enhancement relevant to the project. A project timetable should be used to provide the user with some idea of when staff functions will be affected by system implementation. The project proposal is either accepted or rejected by the user, supervisor, or manager because of it's understood appropriateness in response to the original request.

4. Project Specification. The project proposal accepted for development by the user community will not provide sufficient technical detail to remove all ambiguity for the programming staff. Therefore, information procedures, and environments may be re-written using schemas or pseudocode to better illustrate the software design and hardware configuration. Also, the user community is not directly concerned with the techniques that will be employed to test and debug, nor the procedures that will be used to implement the system under development. These tasks will not be described in the project proposal, but will be included in the formal project specification on which the development will be based.

5. Development. The development work occurs within traditional project life cycles begins once the users have been removed from the evolution of the request by accepting a given project proposal and a formal project specification has been drafted. The work of the development group is dictated by the wording of the project proposal, or the project specification, and not directly influenced by further user input or reaction to the work produced. Development management will often introduce and enforce various policies and formal procedures that will control how development work is accomplished. Typcially, one finds conventions for file naming, locality, and archiving that address data processing auditor concerns, as well as some standards that may dictate coding and testing of programmes that facilitate uniformity of installed work.

6. Implementation. The objective of any project life cycle is to install or implement a new or enhanced procedure or set of procedures. A traditional project life cycle completes itself when tested work is physically moved from the work environment, where all development work occurs, into the agreed environment where users of the completed system are given general access. Traditional methodologies that have been formalized will include defined procedures to accomplish the task of moving completed work and cleaning-up the work environment. Our traditional project life cycle dictates that discrepancies between the completed work and the actual user requirement, that

may have changed since the project proposal was accepted or have been poorly verbalized in the first place, must be addressed with a new project.

Traditional Methodologies Appled to a Non-Traditional Development.

Can the above project life cycle, or any similar traditional project life cycle be effectively applied in a situation where more advanced software tools are applied? The presentation of the paper will become an open forum to identify the tools that are used by the participants, and then follow a traditional project life cycle. The tools and techniques discussed will be applied in a case study manner.

Observations and Conclusion of the Exercise.

The purpose of the presentation of the paper will be to have those participating note observations and draw conclusions based upon the application of fourth generation tools and techniques within a traditional project life cycle.

A Proposed Methodology.

Traditional project life cycles tend to be the most weak in their responsiveness to evolving user requirements, although their primary purpose is to provide a high degree of certainty for the end user and associated auditors. To provide this certainty, a traditional project life cycle sacrifices the ability of the developers to address changes during the development stage of the project life cycle. Fourth generation software and other advanced programming tools and techniques exist to expedite the tasks involved with development work. The current trend in development tools, as evidenced by various prototyping tools and techniques is to not only expedite the project life cycle, but also questions the definition of a project with a clearly defined begin and end. Prototyping is not a methodology in itself, and so requires a mthodology that can recognize the abilities of prototypers to evolve systems with the direct input of an end-user, or group of users.

Implementing a prototyping tool while maintaining a traditional methodology restricts the potential of the tool. A traditional methodology requires a traditional specification. The requirement of a specification to direct a project is not a problem in itself, but the unchangeable nature of a specification does impede the reactiveness of prototype development. The existence of a specification in language that is separate from the prototype is definitely desired, in the same way that a specification that is separate from the source code of any system is desired. It provides a clear declaration of the assumptions of the system designers. What must be reconciled, however, is the ability of a specification to evolve in the same manner that a prototyping tool will enable a system to evolve during it's development.

A traditional development methodology does provide a defined work flow by which productiveness and responsiveness may be measured. More important to the user is the defined timetable by which system implementation may be governed. The users of computer systems have their own staffing and procedural requirements that may have to be co-ordinated with the implementation of automated systems. End-user management also needs to be able to judge the effectiveness of procedures requested and subsequently impelemented in order to make intelligent decisions about future system requirements.

In order to satisfy the realities of programming in a business environment, it is also necessary to be able to deliver completed systems to end-users with which they may process their information with the certainty that that which produced correct results today will produce similar results tomorrow. Any methodology that is used by designers and developers of business systems has to recognize the reality that users cannot function effectively in a system environment that is constantly changing.

The problem in proposing a methodology that enhances today's development software is the reconciliation of the effectiveness and responsiveness that they offer with the necessity to deliver reliable business systems. With no defined development methodology, a project is assured to fail in both effectiveness and depandability.

A Proposed Project Life Cycle

1. Project Initiation

6. Implementation 2. Systems Analysis

5. Prototype Evolution 3. Initial Prototype

4. Systems Specification

A project must have a defined begin and end to provide a sufficient amount of certainty by which users may manage their expectations, staff training, and procedural changes. A defined begin and end also provides the ability to measure the responsiveness of system developers to the requirements of the users of their business systems. The catalyst for system development activity should remain the formal definition of the user's requirement, and the next logical step that follows should be the investigatory work of the systems analyst that assures the user that the designer of the resolution to their request truly understands the request and it's implications to the business as a whole. Project initiation and systems analysis are tasks that should be addressed in the same manner, regardless of whatever tools and techniques are applied to deliver the completed software.

After comprehensive systems analysis work has been completed, the analyst should be free to design a prototype that may be used as a discussion tool with all concerned users, rather than drafting a project proposal. In the same manner that a project proposal is used to form the basis of a technical project specification, the initial prototype of a business system should be documented in such a manner that will clarify all things that have been assumed by the designer and the involved users. The initial prototype is evolved, but each time a new assumption is written into the system, the systems specification is evolved as well. The objective being to deliver a comprehensive prototype that may be given to the user community as a completed system, while maintaining a comprehensive explanation of the evolution that produced the prototype.

Project completion is still defined by the implementation of a prototype that is acceptable to the user as a working business system. User training becomes less of an issue as user involvement is present throughout the development, and the users are thus provided the opportunity to develop their procedures and manage staff training accordingly. Defined project completion also gives a point of reference to measure development effectiveness. An accepted prototype still acts in the same manner as an accepted project proposal in that it terminates one project and requires a new project life cycle to reconcile new requirements. However, the point in time at which the user freezes their requirement becomes sufficiently late in the project life cycle that immediate enhancement caused by new user procedures, policy, or functions is unlikely.

An Approach to Debugging

by Grant W. Fletcher of The Interface Group, Incorporated, and
Kathleen A. Sachara of The Haley Corporation

Abstract of the Paper

A significant amount of effort and, therefore, money is spent debugging systems during the development cycle and after installation.

An Approach to Debugging defines the issue in terms of development and post-installation situations, and then goes on to present a philosophical foundation upon which the matter may be approached to optimize system user/system development efforts.

An Approach to Debugging also presents, in a case study model, some tools and techniques that are applicable to COBOL programming and which may be tailored to other languages.

The objectives of the paper are to present a philosophical approach to debugging that may cause the audience to re-evaluate their own approach to the task, and to share some techniques that have been applied to the task.

Agenda of the Presentation

The What, Why, and When of Debugging.

It is always useful to define the topic at hand, and that is what is addressed by way of the introduction to the paper. The audience is given a definition of what the task is, and some explanation of why and when debugging becomes a necessity.

A Philosophical Foundation.

The success, or failure of a system may depend upon the ability of a system user and a system developer to co-operate during the development cycle and after installation, and that co-operation itself may be dependent upon the philosophy of each participant as it relates to the other. One philosophical approach to the task of debugging that attempts to optimize the user/developer relationship is presented.

Some Tools and Techniques.

The usefulness of knowledge is difficult to quantify until the knowledge itself is evidenced in some material form. The paper presents a case study of some tools and techniques that have been applied to debug an application, both during development and after installation.

The What, Why, and When of Debugging.

Software is usually in a perpetual state of evolution. It is very rare to find a computer programme that has been implemented, and that is not expected to require some modification work in the future. Ideally, we hope that any subsequent modifications to our own software are enhancements that are in response to new user requirements or procedures. Some modification, realistically, occurs in response to user problems encountered with an aspect of the design, coded logic, implementation environment, or manner in which the software is used. Debugging may be viewed as an analytical tool, a technique, or simply the process of addressing the requirements of the latter case.

Webster's defines the term debug as a verb meaning "to find and correct the defects, errors, malfunctioning parts, etc. in". More specifically applied to the job of computer programmers, this definition may be re-written to emphasize the reality that errors may be real or perceived, and that errors occur in computer systems. So, we may now define debugging as the task of finding and correcting the real or perceived errors in computer systems.

The errors that we are commonly required to correct in computer software include errors in design and coded logic, errors in installation environments, and errors related to the usage of the software in question. Their implication may be related to one or more of many factors.

Typically, errors will manifest themselves in the form of corrupted output from a particular programme. Unfortunately, in our more complex systems where data dependancies are numerous, an error may come to surface in a programme that is some number of process steps removed from the offending programme.

Also, in today's business environment, errors can manifest themselves in other ways. Many of our systems have evolved beyond the point of being simple procedural tasks that may be isolated in a few independent computer programmes. Because of the complexity of our systems, and often because of the complexity, necessary, or otherwise, of our computer programmes, user training and the usage of computer programmes themselves may be the source of perceived errors. Perceived errors are more difficult to address as they are often presented as errors relating to programme output. But the perception that an error exists arises because the user's expectation of the output from the programme is not what the programme is intended to produce. If the user's expectations are based upon incorrect assumptions, incomplete training, or improper procedural responses, then an error in user education or software usage may exist. It is as equally important to correct this type of error as those that are directly related to the work done by system developers.

A third manifestation of errors in computer programmes that is becoming more critical in today's business environment, and which should be of particular interest to programmers in an HP3000 (pre Spectrum) environment, is programme performance. Systems may become needlessly overburdened with tasks that may be optimized with better programme or data base design, and a hardware upgrade is simply not available, or not economical. In these situations, if not always, programme performance may be the evidence of errors within the programme's logic, the techniques employed to code the logic, or it may be considered to be the error in itself.

Why errors exist is more ambiguous than the fact that errors do exist, but is of more importance to resolve. With the tools that are available to programmers and other system developers today, it is very easy to write and install computer programmes. But, few, if any of today's tools remove the potential for creating systems that include errors. So, since it is now possible to increase the number of computer programmes produced by one computer programmer during a particular period of time without decreasing the potential for error in each of those programmes being produced, one may argue that the by-product of database management systems, fourth generation languages, and other software advances is the creation of more software errors. This is certainly the case when the causes of errors are not addressed prior to the introduction of some of today's tools.

Errors may be introduced to the computer system early in the project life cycle. One of the first causes of errors in computer systems that eventually require debugging is the communication skills of those people involved with the initial design of a computer programme or system. Poorly communicated requirements, unclear or overly technical proposals, and other misunderstandings can result in systems being developed and implemented that never actually resolve the user's initial request.

Following the project life cycle, the next situation in which errors may be introduced is in the development phase, or during the coding of specific programmes and procedures. Syntax and logic related errors are a very common source of errors in programmes, both during development and after the software has been installed and used.

Another source of errors in programme code that is evidenced more and more today because our systems are often one node in a multi-hardware configuration is the way data is stored within the programme. Once again, communication, now of the intended use of the data, either by designer to programmer, or user to designer, may be the cause of errors detected after programme installation.

Errors that are attributable to programme performance are cases in which the design or technique of coding a particular programme is not the best, or when minor sections of code require inordinant amounts of CPU time to process. These errors can be difficult to detect without prior consideration during the design of the software, and can usually be credited to the level of expertise of the development staff.

Errors may also arise because programmes are developed and used in different account structures, operated under different versions of operating systems, or on completely different CPU's. These types of errors, typically, are the easiest to resolve in a systematic way as each difference between the two environments can be isolated and proven to be the offending factor, or not.

Misconceptions, and improper usage of software are usually the result of inadequate or unclear documentation, or other training factors. When programmes are designed, coded, and installed perfectly there is still the requirement that the programmes be used in the manner, and for the purpose which they were created. Documentation and training can be seen to be the cause of many perceived errors in computer systems. The responsibility to provide a sufficient level of documentation or training for the user has to begin with the designers and developers of computer systems as they are most familiar with the intended usage of the programmes that they implement.

A Philosophical Foundation.

Whether tool, technique, or task, debugging is often the least scientific element of our function as computer programmers. It is common to hear debugging described as an art, or some other subjective process, and it is too often found that the task is delegated to our most junior staff and rationalized as an important learning experience for them to endure. Rarely are programmes or systems designed with any direct consideration for the fact that the project will undergo some level of debugging during it's life cycle, and that the implemented work from the project will itself be the subject of some future debugging, or at least investigation of it's processing or performance.

The reasons discussed for why and when debugging occurs have one common element, and that one element contributes to the ambiguity and subjectiveness found relative to the task. The ability of the user to communicate their requirement to the system designer is the first potential cause of errors in computer programmes. Then, the ability of the designer to communicate the intent and specification of the programme to the programmer, the ability of the programmer to describe and document the work that has been done, and the ability of the user to understand the use of the implemented programme all compound the potential for errors in the systems that we develop. Debugging becomes a difficult task because the interpretation of each participant in the project of what the project is addressing is rarely similar.

But, there is no good reason why debugging cannot be made easier with a different approach to the matter. Computer scientists are the rare exception within the scientific community who appear to assume that their work will be functional and error free on the first attempt, and rarely design for the possibility that is not. Most computer programmes are written under the philosophy that testing will isolate any and all errors, and that the errors that do occur are unique enough that they cannot be generalized. Perceived errors are rarely considered to be within the realm of responsibility of computer programmers, but usually considered to be problems that the users themselves must resolve.

Approaching system development with the idea that debugging is a design issue, rather than an ad hoc activity to be considered if and when problems arise, causes programmes to be written that are easier to debug. Designing programmes to be debugged should be a formal criteria for all systems developed within a structured programming environment. Where possible, fourth generation tools should be evaluated for their ability to permit the programmer to design debugging considerations into programmes. Programmes designed and written to be debugged are generally the simplest to debug.

Implemented within structured programming guidelines, but acceptable within any standardized programming environment, this approach to programme development inherently leads to a definable and systematic approach to the resolution of any error. An informal, but equally recognizable aspect of the philosophy of designing programmes for debugging is that debugging should occur within a defined framework, providing both the system developer and the system user with a common understanding of how the task will be undertaken.

Once the task of debugging is placed within a defined structure, then the procedures and techniques for investigation may not only be described to the user community, but designed to be employed by the users themselves. Giving the user sufficient insight to the workings of computer processes that are executing tasks on their behalf reduces the possibility of perceived errors, and strengthens the understanding between the system user and the system developer at all stages of system development. Increasing the direct involvement of the user in the debugging process also reduces the further potential for misunderstanding during the initial stage of debugging that involves identifying the problem and describing it in terms that are common to both the user and the programmer.

Philosophically, and often for practical reasons, the task of debugging should begin with the programme input and output that the user perceives to be in error. Re-compilation of source code should not be necessary until the cause of the problem is clear, and the resolution has been written into the programme. Meeting this criteria demands that the implemented programme include designed logic to facilitate the gathering of all information necessary to isolate any process of the programme that may be in error. Preferrably, this logic should be conditional upon parameters passed during the execution of the programme.

The composition of the above elements presents one philosophy on debugging computer programmes that places emphasis on formalizing debugging considerations in the design of computer systems. Unfortunately, this does not directly address the potential for design errors themselves, but, since the system developer/system user relationship is enhanced through more direct user involvement in the debugging process, the potential is reduced indirectly. The introduction of debugging in the design of computer systems also improves the understanding amoung all concerned parties of the intent and use of the systems or programmes implemented.

Some Tools and Techniques.

The philosophical approach to debugging described above postulates that debugging considerations are a critical element of programme design. This section of the paper presents, by way of example, some tools and techniques that help to implement a design based approach to debugging that also reduces the need to recompile source code during the investigatory stage of debugging.

The example that follows is presented within the context of a COBOL programme with some called routines written in SPL. Many, if not all of the techniques illustrated may be implemented with other programming languages.

The discussion will evolve around the following programme as participants develop a given case study that will be included with the presentation materials. The following tools and techniques will be illustrated during the discussion to give the participants some insight to their potential uses.

MPE DEBUG,
Debugging mode in Cobol,
Job Control Words,
Copy Libraries,
Segmentation, and
Process timing techniques.

```
$Control  nolist,   source,   warn,   map,   code&
$     ,   bounds,   crossref,   locking,   mixed&
$     ,   verbs, quote=', uslinit
$Set X9=off
*     Off=Test Compilation, On=Implementation Compilation.
$Set X0=on
*     Off=SPL, On=Cobol.
$IF X9=off
$Control DEBUG
$IF
 Identification Division.
 Program-id.
      Pgm-Manager.
 Remarks.
      Summary

      Usage

      Installation

      Operation
           RUN pgm; parm=
                0, or not defined is normal operation.
                1, is 'process-debug' (process-activity-debug).
                64, is 'not UPDATING' (UPDATE-SWITCH).
           Input
           Processing
           Output
           Recovery/Debugging

 Environment Division.
 Copy A20b200 of A20bLIB nolist.
 Data Division.
 File Section.
 Working-Storage Section.
 Copy A20b300 of A20bLIB nolist
      replacing =='$Title                                  '== by
                =='$Actual Title                           '==.
 Procedure Division.
 Copy A20b600 of A20bLIB nolist.
 Function-Process.
      Display 'Virgin programme, no functions defined.'
           upon sysout.
*Include functional procedure files.
 Copy A20b800 of A20bLIB nolist.
$Control    list
*Segmentation.
*     Capability IA, PH are required.
$Control  nolist
 End-of-programme Section 99.
```

```
Configuration Section.                                             A20B200
Source-computer.  HP300048 with debugging mode.                    A20B200
Object-computer.  HP3000xx.                                        A20B200
Special-names.                                                     A20B200
     CONDITION-CODE is istatus,                                    A20B200
     SW9 is UPDATE-SWITCH, off status is UPDATING,                 A20B200
     TOP is PAGE-EJECT.                                            A20B200
```

```
*Programme.                                                             A20B300
 01 Programme pic x(28) value spaces.                                   A20B300
 01 Father-pin pic s9(4) comp value 0.                                  A20B300
    88 father-process value 0.                                          A20B300
 01 process-control.                                                    A20B300
    11 process-activity pic x(3) value spaces.                          A20B300
       88 process-end value 'END'.                                      A20B300
    11 process-activity-debug pic x(1) value spaces.                    A20B300
       88 process-debug value '1'.                                      A20B300
 01 flag-dialogue pic s9(4) comp value 0.                               A20B300
    88 process-dialogue value 1.                                        A20B300
 01 process-error pic s9(4) comp value 0.                               A20B300
    88 no-process-error value 0.                                        A20B300
*Programme Title.                                                       A20B300
 01 Programme-Name.                                                     A20B300
    11 Title pic x(40) value                                            A20B300
       '$Title                                  '.                      A20B300
    11 filler pic x(26) value spaces.                                   A20B300
    11 programme-date pic x(8) value spaces.                            A20B300
    11 filler pic x(1) value spaces.                                    A20B300
    11 programme-time pic x(5) value spaces.                            A20B300
                                                                        A20B300
*Processing Variables.                                                  A20B300
 01 fstatus pic x(2) value spaces.                                      A20B300
    88 fstatus-ok value '00'.                                           A20B300
 01 lgth pic s9(4) comp value 0.                                        A20B300
 01 mpe-error pic s9(4) comp value 0.                                   A20B300
 01 number-out pic 9(5)- value zeroes.                                  A20B300
 01 numchar pic s9(4) comp value 0.                                     A20B300
 01 passed-info pic x(80) value spaces.                                 A20B300
 01 passed-parm pic s9(4) comp value 0.                                 A20B300
 01 pin pic s9(4) comp value 0.                                         A20B300
 01 work.                                                               A20B300
    11 work-1 pic s9(9) comp value 0.                                   A20B300
    11 work-2 pic s9(9) comp value 0.                                   A20B300
    11 work-3 pic s9(9) comp value 0.                                   A20B300
*CreateProcess Variables.                                               A20B300
 01 process-items.                                                      A20B300
    11 items pic s9(4) comp occurs 13 times.                            A20B300
 01 process-itemvalues.                                                 A20B300
    11 itemvalue-1 pic x(2) value spaces.                               A20B300
    11 itemvalue-2 pic x(2) value spaces.                               A20B300
    11 itemvalue-3 pic x(2) value spaces.                               A20B300
    11 itemvalue-4 pic x(2) value spaces.                               A20B300
    11 itemvalue-5 pic x(2) value spaces.                               A20B300
    11 itemvalue-6 pic x(2) value spaces.                               A20B300
    11 itemvalue-7 pic x(2) value spaces.                               A20B300
    11 itemvalue-8 pic x(2) value spaces.                               A20B300
    11 itemvalue-9 pic x(2) value spaces.                               A20B300
    11 itemvalue-10 pic x(2) value spaces.                              A20B300
    11 itemvalue-11 pic x(2) value spaces.                              A20B300
    11 itemvalue-12 pic x(2) value spaces.                              A20B300
    11 itemvalue-13 pic x(2) value spaces.                              A20B300
```

```
*Date Variables.                                                  A20B300
 01 work-dates.                                                   A20B300
    11 work-date-day pic s9(4) comp value 0.                      A20B300
    11 work-date-julian pic s9(9) comp value 0.                   A20B300
    11 work-date-yyyymmdd.                                        A20B300
       21 work-date-century pic x(2) value spaces.                A20B300
       21 work-date-yymmdd.                                       A20B300
          31 work-date-yy pic 9(2) value zeroes.                  A20B300
          31 work-date-mm pic 9(2) value zeroes.                  A20B300
          31 work-date-dd pic 9(2) value zeroes.                  A20B300
*Timer Variables.                                                 A20B300
 01 timer-1 pic s9(9) comp value 0.                               A20B300
 01 timer-2 pic s9(9) comp value 0.                               A20B300
 01 timer-debug-1 pic s9(9) comp value 0.                         A20B300
 01 timer-debug-2 pic s9(9) comp value 0.                         A20B300
 01 timer-out pic s9(9) sign leading separate.                    A20B300
 01 work-time.                                                    A20B300
    11 work-time-hh pic 9(2) value zeroes.                        A20B300
    11 work-time-mm pic 9(2) value zeroes.                        A20B300
    11 work-time-ss pic 9(2) value zeroes.                        A20B300
                                                                  A20B300
*Input.                                                           A20B300
 01 stdinx pic s9(4) comp value 0.                                A20B300
 01 function-input.                                               A20B300
    11 function-type pic x(1) value spaces.                       A20B300
    11 function-command pic x(79) value spaces.                   A20B300
                                                                  A20B300
*Output.                                                          A20B300
 01 stdlist pic s9(4) comp value 0.                               A20B300
 01 clear-screen.                                                 A20B300
    11 filler pic x(1) value %33.                                 A20B300
    11 filler pic x(1) value 'h'.                                 A20B300
    11 filler pic x(1) value %33.                                 A20B300
    11 filler pic x(1) value 'J'.                                 A20B300
 01 cr pic x(1) value %15.                                        A20B300
 01 end-continue.                                                 A20B300
    11 filler pic x(1) value %15.                                 A20B300
    11 filler pic x(1) value %12.                                 A20B300
    11 filler pic x(38) value                                     A20B300
       '     Please hit "Return" to continue.;'.                  A20B300
 01 error-messages.                                               A20B300
    11 filler pic x(50) value                                     A20B300
       'Error, can not open $STDINX, or $STDLIST          '.      A20B300
    11 filler pic x(50) value                                     A20B300
       'Error, JCW or CIERROR not zero                    '.      A20B300
    11 filler pic x(50) value                                     A20B300
       'Error, programme jcw not zero                     '.      A20B300
    11 filler pic x(50) value                                     A20B300
       'Error, software license is not valid              '.      A20B300
 01 errormessages redefines error-messages.                       A20B300
    11 error-message pic x(50) occurs 4 times.                    A20B300
 01 error-limit pic s9(4) comp value 4.                           A20B300
 01 format-off.                                                   A20B300
    11 filler pic x(1) value %33.                                 A20B300
    11 filler pic x(1) value 'X'.                                 A20B300
```

```
01 lf pic x(1) value %12.                                         A20B300
01 message-buffer pic x(160) value spaces.                        A20B300
01 prompt-function.                                               A20B300
   11 filler pic x(1) value %33.                                  A20B300
   11 filler pic x(7) value '&a2r00C'.                            A20B300
   11 filler pic x(9) value 'Function?'.                          A20B300
```

```
Declaratives.                                                    A20B600
Debugger Section 01.  Use for debugging on all procedures.       A20B600
Debug-Dump.                                                      A20B600
     Display                                                     A20B600
          '                    ',                                A20B600
          '* * * * *  D e b u g  D u m p  * * * * *'.            A20B600
     Display                                                     A20B600
          '                    ',                                A20B600
          '         ======================         '.            A20B600
     Display DEBUG-ITEM, cr, lf.                                 A20B600
Debug-Timer.                                                     A20B600
     Call intrinsic "PROCTIME" giving timer-debug-1.             A20B600
     IF timer-debug-2 = 0 then                                   A20B600
          move '1' to process-activity-debug                     A20B600
          call intrinsic "GETINFO"                               A20B600
               using passed-info, \\, passed-parm                A20B600
               giving mpe-error                                  A20B600
          move passed-parm to number-out                         A20B600
          display                                                A20B600
               'Information Passed', cr, lf, passed-info         A20B600
               , 'Parm = ', number-out                           A20B600
     ELSE                                                        A20B600
          compute timer-out = timer-debug-1 - timer-debug-2      A20B600
          display                                                A20B600
               '      '                                          A20B600
               , CURRENT-DATE                                    A20B600
               , ' '                                             A20B600
               , TIME-OF-DAY                                     A20B600
               , ' CPU time (milliseconds) since last point '    A20B600
               , timer-out.                                      A20B600
     Compute timer-debug-2 = timer-debug-1.                      A20B600
Debug-Dump-End.                                                  A20B600
     Display                                                     A20B600
          '========================================'             A20B600
          , '========================================'.          A20B600
     IF process-dialogue then                                    A20B600
          Display 'Debug end - please hit "Return" to continue.' A20B600
          Accept pin                                             A20B600
          Compute pin = 0.                                       A20B600
End Declaratives.                                                A20B600
```

```
Pgm-Main Section 11.                                             A20B600
Perform Initializations.                                         A20B600
IF no-process-error then perform Processes until process-end     A20B600
ELSE next sentence.                                              A20B600
Perform Conclusions.                                             A20B600
GOBACK.                                                          A20B600
                                                                 A20B600
Initializations.                                                 A20B600
     Move CURRENT-DATE to programme-date.                        A20B600
     Move TIME-OF-DAY to work-time.                              A20B600
     String work-time-hh delimited by size                       A20B600
          , ":" delimited by size                                A20B600
          , work-time-mm delimited by size                       A20B600
          into programme-time                                    A20B600
     Call intrinsic "PROCINFO"                                   A20B600
          using lgth, numchar, \0\, \10\, programme.             A20B600
     Call intrinsic "FOPEN" using \\, \%254\ giving stdinx.      A20B600
     IF istatus = 0 then next sentence ELSE compute stdinx = 0.  A20B600
     Call intrinsic "FOPEN"                                      A20B600
          using \\, \%214\, \%1\ giving stdlist.                 A20B600
     IF istatus = 0 then next sentence ELSE compute stdlist = 0. A20B600
     IF stdinx = 0 OR stdlist = 0 then compute process-error = 1.A20B600
     String format-off delimited by size                         A20B600
          , clear-screen delimited by size                       A20B600
          , programme-name delimited by size                     A20B600
          into message-buffer.                                   A20B600
     Call intrinsic "PRINT" using message-buffer, \-86\, \%40\.  A20B600
     IF no-process-error then                                    A20B600
          call intrinsic "GETJCW" giving mpe-error               A20B600
          IF mpe-error = 0 then                                  A20B600
               move 'CIERROR.' to message-buffer                 A20B600
               call intrinsic "FINDJCW"                          A20B600
                    using message-buffer, pin, mpe-error         A20B600
               IF mpe-error = 0 and pin = 0 then                 A20B600
                    call intrinsic "FINDJCW"                     A20B600
                         using programme, pin, mpe-error         A20B600
                    IF mpe-error = 3                             A20B600
                    OR (mpe-error = 0 and pin = 0) then          A20B600
                         call "PROCESS'PROFILE"                  A20B600
                              using \stdinx\, flag-dialogue      A20B600
                         call intrinsic "FATHER" using father-pinA20B600
                         IF istatus = 0 then NEXT SENTENCE       A20B600
                         ELSE compute father-pin = 0             A20B600
                    ELSE compute process-error = 3               A20B600
               ELSE compute process-error = 2                    A20B600
          ELSE compute process-error = 2                         A20B600
     ELSE next sentence.                                         A20B600
```

```
Conclusions.                                                    A20B600
     Move CURRENT-DATE to programme-date.                       A20B600
     Move TIME-OF-DAY to work-time.                             A20B600
     String work-time-hh delimited by size                      A20B600
          , ":" delimited by size                               A20B600
          , work-time-mm delimited by size                      A20B600
          into programme-time.                                  A20B600
     Call intrinsic "PUTJCW"                                    A20B600
          using programme, process-error, mpe-error.            A20B600
     IF no-process-error then                                   A20B600
          display                                               A20B600
               cr, lf, programme-date, ' ', programme-time, '. 'A20B600
               , 'Normal end of programme.' UPON SYSOUT         A20B600
     ELSE                                                       A20B600
          move process-error to number-out                      A20B600
          string "Programme Error " delimited by size           A20B600
               , number-out delimited by size                   A20B600
               , "." delimited by size                          A20B600
               into message-buffer                              A20B600
          call intrinsic "PRINT"                                A20B600
               using message-buffer, \-22\, \%40\               A20B600
          IF process-error < 0                                  A20B600
          OR process-error > error-limit then                   A20B600
               Display                                          A20B600
                    cr, lf, programme-date                      A20B600
                    , ' ', programme-time, '.  '                A20B600
                    , 'Abnormal end of programme.' UPON SYSOUT  A20B600
               compute process-error = %100000                  A20B600
               call intrinsic "SETJCW" using process-error      A20B600
          ELSE                                                  A20B600
               Display                                          A20B600
                    cr, lf, error-message(process-error)        A20B600
                    , cr, lf, programme-date                    A20B600
                    , ' ', programme-time, '.  '                A20B600
                    , 'Abnormal end of programme.' UPON SYSOUT. A20B600
```

```
Processes.                                                       A20B600
     Move spaces to function-input.                              A20B600
     Move CURRENT-DATE to programme-date.                        A20B600
     Move TIME-OF-DAY to work-time.                              A20B600
     String work-time-hh delimited by size                       A20B600
          , ":" delimited by size                                A20B600
          , work-time-mm delimited by size                       A20B600
          into programme-time                                    A20B600
     String format-off delimited by size                         A20B600
          , clear-screen delimited by size                       A20B600
          , programme-name delimited by size                     A20B600
          , prompt-function delimited by size                    A20B600
          into message-buffer.                                   A20B600
     Call "INPUT" using message-buffer, function-input, \80\.    A20B600
     IF function-input = spaces then                             A20B600
          call intrinsic "FATHER" giving father-pin              A20B600
          IF istatus = 0 then                                    A20B600
               call intrinsic "ACTIVATE" using \0\, \3\          A20B600
          ELSE move 'END' to process-control                     A20B600
     ELSE                                                        A20B600
          call "EDIT'UP" using function-input, \80\              A20B600
          IF function-input = 'EXIT' OR 'END' then               A20B600
               move 'END' to process-control                     A20B600
          ELSE                                                   A20B600
               IF function-type = ':' then                       A20B600
                    perform Command                              A20B600
                    call "INPUT'0" using end-continue, pin, \1\  A20B600
               ELSE                                              A20B600
                    IF function-type = '$' then                  A20B600
                         perform CreateProcess                   A20B600
                         call "INPUT'0"                          A20B600
                              using end-continue, pin, \1\       A20B600
                    ELSE                                         A20B600
                         perform Function-Process                A20B600
                         IF no-process-error then                A20B600
                              call "INPUT'0"                     A20B600
                              using end-continue, pin, \1\       A20B600
                         ELSE                                    A20B600
                              move process-error to number-out   A20B600
                              compute process-error = 0          A20B600
                              string                             A20B600
                                   "     Process Error "         A20B600
                                        delimited by size        A20B600
                                   , number-out                  A20B600
                                        delimited by size        A20B600
                                   , "." delimited by size       A20B600
                                   into message-buffer           A20B600
                              call intrinsic "PRINT"             A20B600
                                   using message-buffer          A20B600
                                        , \-25\, \%40\           A20B600
                              call "INPUT'0"                     A20B600
                                   using end-continue, pin, \1\. A20B600
```

```
Command.                                                         A20B600
     Move spaces to message-buffer.                              A20B600
     String function-command delimited by size                   A20B600
          , cr delimited by size                                 A20B600
          into message-buffer.                                   A20B600
     Call intrinsic "COMMAND" using                              A20B600
          message-buffer, mpe-error, pin.                        A20B600
     IF mpe-error = 0 then NEXT SENTENCE                         A20B600
     ELSE                                                        A20B600
          move mpe-error to number-out                           A20B600
          IF mpe-error < 0 then                                  A20B600
               string "Command Warning " delimited by size       A20B600
                    , number-out delimited by size               A20B600
                    into message-buffer                          A20B600
               call intrinsic "PRINT"                            A20B600
                    using message-buffer, \-21\, \%40\           A20B600
          ELSE                                                   A20B600
               string "Command Error " delimited by size         A20B600
                    , number-out delimited by size               A20B600
                    into message-buffer                          A20B600
               call intrinsic "PRINT"                            A20B600
                    using message-buffer, \-19\, \%40\.          A20B600
     Move spaces to message-buffer, function-input.              A20B600
     Compute mpe-error = 0.                                      A20B600
     Compute pin = 0.                                            A20B600
```

```
CreateProcess.                                              A20B600
    Move spaces to message-buffer.                          A20B600
    Move function-command to message-buffer.                A20B600
    Compute items(1) = 10.                                  A20B600
    Compute items(2) = 6.                                   A20B600
    Compute items(3) = 3.                                   A20B600
    Compute items(4) = 0.                                   A20B600
    Move %3 to itemvalue-1.                                 A20B600
    Move %75000 to itemvalue-2.                             A20B600
    Move %41 to itemvalue-3.                                A20B600
    Move 0 to itemvalue-4.                                  A20B600
    Call intrinsic "CREATEPROCESS"                          A20B600
        using mpe-error, pin, message-buffer                A20B600
        , process-items, process-itemvalues.                A20B600
    IF mpe-error = 0 then NEXT SENTENCE                     A20B600
    ELSE                                                    A20B600
        move mpe-error to number-out                        A20B600
        IF mpe-error < 0 then                               A20B600
            string "Loader Warning " delimited by size      A20B600
                , number-out delimited by size              A20B600
                into message-buffer                         A20B600
            call intrinsic "PRINT"                          A20B600
                using message-buffer, \-20\, \40\           A20B600
        ELSE                                                A20B600
            string "Loader Error " delimited by size        A20B600
                , number-out delimited by size              A20B600
                into message-buffer                         A20B600
            call intrinsic "PRINT"                          A20B600
                using message-buffer, \-18\, \%40\.         A20B600
    Call intrinsic "KILL" using pin.                        A20B600
    Move spaces to message-buffer, function-input.          A20B600
    Compute mpe-error = 0.                                  A20B600
    Compute pin = 0.                                        A20B600
```

```
Subroutines Section 91.                                          A20B800
SUBROUTINE-Start-Clock.                                          A20B800
     Call intrinsic "PROCTIME" giving timer-1.                   A20B800
     Display                                                     A20B800
          "Timer     "                                           A20B800
          , CURRENT-DATE                                         A20B800
          , " "                                                  A20B800
          , TIME-OF-DAY                                          A20B800
          , " Clock Started.".                                   A20B800
                                                                 A20B800
SUBROUTINE-Stop-Clock.                                           A20B800
     Call intrinsic "PROCTIME" giving timer-2.                   A20B800
     Compute timer-out = timer-1 - timer-2.                      A20B800
     Display                                                     A20B800
          "Timer     "                                           A20B800
          , CURRENT-DATE                                         A20B800
          , " "                                                  A20B800
          , TIME-OF-DAY                                          A20B800
          , " Clock Stopped: CPU used (milliseconds) "           A20B800
          , timer-out.                                           A20B800
```

```
Procedure Process'Profile(stdinx,flag);
     Value   stdinx;
     Integer stdinx, flag;
!This procedure defines whether programmatic dialogue may occur.
BEGIN
INTRINSIC FFILEINFO, WHO, PRINTFILEINFO;
Logical array  mode(0:0);
Byte array     mode'ba(*)=mode;

!Input the user's profile.
WHO(mode);
IF mode.(12:2)=%1<<session>> then BEGIN
     FFILEINFO(stdinx,2,mode'ba);
!If it is recognized that the user is operating a session, and that
!stdinx (opened in Initializations) is either $STDIN or $STDINX, then
!the programme may expect the user to respond to prompting.  Otherwise,
!the user is passing input into the programme using a file, and it is
!logically incorrect to expect the user to respond to alternate action
!prompting (from a recoverable exception).
     IF %4<=INTEGER(mode.(10:3))<=%5 then flag:=1
     ELSE flag:=0;
END
ELSE flag:=0;
END<<Process'Profile>>;
```

```
Procedure Input(parameter0,parameter1,parameter2);
     Value parameter2; Integer parameter2;
     Array parameter0<<0:79>>, parameter1;
<<
Summary        : This procedure outputs the prompt, ending with "?", in
                 parameter 0, and returns the input user response in
                 parameter 1.  The length of the expected response is
                 passed in parameter 2.
>>
BEGIN
INTRINSIC PRINT, READX;

Byte array     input'ba(*)=parameter1;
Array          message(0:79);
     Byte array     message'ba(*)=message;
Byte array     prompt(*)=parameter0;
Integer        lgth:=0;

lgth:=SCAN prompt UNTIL "?";
message:="  ";
MOVE message(1):=message,(79);
MOVE message'ba:=prompt,(lgth+1);
PRINT(message,-(lgth+2),%320);
input'ba:=" ";
MOVE input'ba(1):=input'ba,(parameter2);
lgth:=READX(parameter1,-(parameter2+1));
END<<Input>>;

Procedure Input'0(parameter0,parameter1,parameter2);
     Value parameter2; Integer parameter2;
     Array parameter0<<0:79>>, parameter1;
<<
Summary        : This procedure outputs the prompt, delimited by ";",
                 in parameter 0, and returns the input user response in
                 parameter 1.  The expected length of the user response
                 is passed in parameter 2.
>>
BEGIN
INTRINSIC PRINT, READX;

Byte array     input'ba(*)=parameter1;
Array          message(0:79);
     Byte array     message'ba(*)=message;
Byte array     prompt(*)=parameter0;
Integer        lgth:=0;

lgth:=SCAN prompt UNTIL ";";
message:="  ";
MOVE message(1):=message,(79);
MOVE message'ba:=prompt,(lgth);
PRINT(message,-(lgth+1),%320);
input'ba:=" ";
MOVE input'ba(1):=input'ba,(parameter2);
lgth:=READX(parameter1,-(parameter2+1));
END<<Input'0>>;
```

```
Procedure Edit'Up(parameter0,parameter1);
     Value parameter1; Integer parameter1;
     Array parameter0;
<<
Summary       : This procedure edits a string of characters passed in
                parameter 0 for a length of characters defined in
                parameter 1.  Any lower case letters are shifted up.
>>
BEGIN
Byte array buffer'ba(*)=parameter0;
parameter1:=parameter1 - 1;
DO BEGIN
     IF %141<=INTEGER(buffer'ba(parameter1))<=%172 then
          buffer'ba(parameter1):=buffer'ba(parameter1) - %40;
     parameter1:=parameter1 - 1;
END UNTIL parameter1 < 0;
END<<Edit'Up>>;
```

Optimizing IMAGE/TurboIMAGE Blocking and Buffering

by David Merit, Bradmark Computer Systems, Inc.

Introduction

Somewhere beyond the performance considerations of IMAGE and TurboIMAGE database design everyone knows--integer-keyed masters, sorted paths, and so forth--are the important fundamentals of blocking and buffering. These rudiments are less known and therefore often ignored, but they are significant in determining overall database performance and disc utilization.

For our purposes here, blocking involves the division of data storage on disc and its transfer to memory; buffering concerns the storage of data in memory in IMAGE/TurboIMAGE's internal control block.

Blocking and buffering are important not only for new databases but also for existing ones, and modifications normally do not impact existing programs (non-programmers and users of third-party "canned" software take heart).

Like all things tunable, changes in blocking or buffering can make things better, worse, or the same, and there are no easy answers to the questions about what to do. Every change involves a trade-off, with the type, frequency, and volume of database access, the database size, structure, and complexity, system configuration, and version of IMAGE/TurboIMAGE all significant factors in determining what needs to be done.

This article is, therefore, not an attempt to provide definitive "yes/no" answers. It attempts to do something better: equip you with the knowledge required to reach your own correct conclusions about optimizing blocking and buffering in your databases.

The block

The block is the basic unit of storage for a dataset. Each disc I/O against a database consists of a block--no less, no more (irrespective of disc caching). Each DBGET against a dataset reads not only the entry being "gotten" into the internal buffers in memory but its entire block; a DBPUT, DBDELETE, and DBUPDATE reads in the block that contains the desired entry, updates it, and writes it back.

The amount of data that will fit into a block is determined by the block size, which represents the length of a disc block and is equal to the record length of the MPE file that contains the dataset. In fact, as far as MPE is concerned, a dataset disc block (which probably contains several entries) is just one long record. A database block always begins and ends on a sector boundary and since a sector is 128 words, the block size must be a multiple of 128 words (256, 384, 512, 640, and so forth).

Blocking factor

The blocking factor represents the number of entries which are contained in each block, and therefore the number of entries gotten with each I/O and transferred into the buffers.

This blocking factor is internal to IMAGE/TurboIMAGE and should not be confused with the blocking factor of the MPE file that contains the dataset, which is always 1.

BLOCKMAX

The block size and blocking factor are automatically calculated by DBSCHEMA.PUB.SYS based on the BLOCKMAX declared in the beginning of a schema with a **$CONTROL BLOCKMAX** command. The BLOCKMAX sets the upper limit for the block sizes of all the datasets in the database, which does not mean that DBSCHEMA will set the block size for every dataset equal to the BLOCKMAX; rather, it will not exceed it and will assign a smaller block size to a dataset if this results in more efficient disc utilization.

By default, the BLOCKMAX is 512 words; if no **$CONTROL BLOCKMAX** command is specified, the default is assigned. The lowest allowable BLOCKMAX is 128 words, the highest is 2048 words for IMAGE and 2560 words for TurboIMAGE.

DBSCHEMA assigns each dataset's blocking factor based on its block size by dividing the block size by the media record length (the length of the data entry plus pointers).

Bit map

DBSCHEMA makes sure this blocking factor leaves room in the block for a bit map of between 1 and 16 words, the length depending on the blocking factor.

The bit map, used to keep track of free space within each block, is located at the beginning of the block. It occupies one word of disc space for every blocking factor multiple of 16 (since a word consists of 16 bits), so blocking factors of 1 to 16 require one word, 17 to 32 require two words, and so forth. The maximum bit map size is 16 words (since the maximum blocking factor is 255). If there is not enough space for the bit map, DBSCHEMA reduces the blocking factor as required.

DBSCHEMA

So, the BLOCKMAX, it seems is the only control you have in determining the disc storage of your database--DBSCHEMA takes it all from there: It calculates the block size for each dataset and its resultant blocking factor.

This calculation would be fine if DBSCHEMA did a good job of it, but DBSCHEMA is bent on optimizing disc usage at the expense of throughput. As mentioned, DBSCHEMA will assign a lower-than-BLOCKMAX block size if this results in less wasted disc space than that wasted if the BLOCKMAX had been assigned. Unfortunately, a lower block size usually means a lower blocking factor, and a lower blocking factor means that less entries are contained in each block. So more blocks must be read to get the same number of entries.

For example, for a dataset with a media entry length (data + pointers) of 63 words, DBSCHEMA would assign a blocking factor of six, which results in a block size of 384 words with five words wasted (63 * 6 + 1 = 379) - even if the database BLOCKMAX is 512 words. It would be more I/O efficient, although slightly less disc efficient, to instead size the block at 512 words, which yields a blocking factor of eight with seven words wasted (63 * 8 + 1 = 505).

A less obvious circumstance in which DBSCHEMA's assigned blocking factor may not be the best is in datasets which are created with very low capacities, since DBSCHEMA will not assign a blocking factor higher than the capacity. The problem becomes apparent when you increase the capacity but retain the existing blocking factor.

Assigning blocking factors

It is normally preferable to trade off some disc space for a higher blocking factor, since this improves throughput, which is more costly than disc space. Usually, the disc savings from lower-than-BLOCKMAX block sizes are not very significant.

Fortunately, it is possible to override DBSCHEMA's calculation and assign your own blocking factors, as you would want to do in the noted scenarios. You can do this for all the datasets in the database or only for particular datasets by enclosing the blocking factor in parentheses on the dataset's **CAPACITY** line in the schema; for example:

```
     CAPACITY:  100000(8);
```

This does not mean you have to recalculate and declare blocking factors for every dataset in your database--you need to override only those which result in datasets blocked less than the BLOCKMAX.

For an existing database, do a **:LISTF,2** to identify the lower-than-BLOCKMAX sets. Look at the record lengths of the dataset files--those which are lower than the BLOCKMAX are candidates for reblocking.

For a new database, add a **$CONTROL NOROOT** at the top of the schema which suppresses the creation of the root file but displays the dataset summary table at the bottom, as shown in this example from HP's TurboIMAGE Reference Manual:

DATA SET NAME	TYPE	FLD CNT	PT CT	ENTR LGTH	MED REC	CAPACITY	BLK FAC	BLK LGTH	DISC SPACE
EMPLOYEE	M	4	1	7	17	500	30	512	72
PROJECT-MASTER	M	2	1	10	20	75	19	382	15
LABOR	D	4	2	10	18	10024	28	506	1436

Here, the EMPLOYEE set is blocked perfectly--a block length (BLK LGTH) of 512 is an even multiple of 128, so there is no wasted space. DBSCHEMA arrived at this figure by multiplying the media record length (MED REC) by the blocking factor (BLK FAC) and then added two words for the bit map. The LABOR set's block length is 506 words which when fit into a block size of 512 words (the nearest 128-word multiple), wastes only 6 words per block (512 - 506 = 6).

For PROJECT-MASTER, DBSCHEMA is saving disc space--it chose a buffer length of 382 words because it is very disc efficient, since it fits snugly into a lower-than-BLOCKMAX block size of 384 words (384 - 382 = 2 words wasted per block). This results in a blocking factor of 19. This dataset could be reblocked into a 512 word block, which would waste 10 words per block instead of two, but would increase the blocking factor from 19 to 25 (20 * 25 = 500 + 2 word bit map = 502).

So, to determine for which sets DBSCHEMA assigns lower blocking factors, compare the BLK LGTH with the BLOCKMAX. Those which fall short of the BLOCKMAX by more than one media entry length (MED REC) should be examined to determine whether the blocking factor could be increased--thus increasing the block size--while staying within the BLOCKMAX.

In doing your calculation, make sure to use the media entry length instead of the data entry length (ENTR LGTH), since the data entry length does not include the pointers. Also remember that an increased blocking factor may require a larger bit map. Make sure there is sufficient space in the block.

Effective blocking by design

It is a good idea to keep blocking effectiveness in mind when designing a database, since it is quite easy to design a dataset which cannot be blocked efficiently.

When designing a database it is best to avoid very long records since they result in very low blocking factors and may require use of a large BLOCKMAX (even up to the IMAGE maximum of 2048 words or TurboIMAGE maximum of 2560 words) to get a decent blocking factor.

It can be tempting to pack something like a customer or part record full of so much data that these datasets have very long record lengths and therefore blocking factors of only 1 or 2. Normally, it is those datasets that are heavily accessed and in which you can ill-afford performance problems. The ramifications are worsened for master sets because the impact of secondaries is greater in masters with low blocking factors.

Another problem is that some datasets just cannot be effectively blocked; for example, a media entry length of 260 words will never block efficiently since it cannot fit into a multiple of 128 without leaving a lot of residual space. If, on the other hand, a media entry length of 254 could be used, it and a one- word bit map would fit quite snugly into any valid block size.

This, of course, does not suggest that you should seriously compromise your database structure, but rather that there is often enough flexibility so you can bend into near-128 word multiples.

Again, remember that you must calculate blocking factors based on media entry length--not data entry length. To determine the media entry length, either process your schema through DBSCHEMA and look at the table at the end or add in the amount of overhead to the data entry length as follows:

master (IMAGE): 5 words, plus 5 words per path
master (TurboIMAGE): 5 words, plus 6 words per path
detail: 4 words per path

Also, be sure to leave room for the bit map. In the example, if the media entry length was 256 words, the extra word needed for the bit map would force the block size up by a sector--128 words--of which only 1 word would be used!

DBCONV

Actually, this scratches the surface of something which caused a quick, painless, and relatively covert loss of blocking effectiveness in thousands of HP3000 sites--the conversion to TurboIMAGE via DBCONV.PUB.SYS.

As we've seen, the media entry length is the basis for calculating a blocking factor, so a change in the media entry length will usually require a change in the blocking factor. Database restructuring utilities are smart enough to recalculate the blocking factor and assign it for datasets in which an item length is changed or an item or path is added or deleted. A structural change via DBUNLOAD/DBLOAD will also result in a recalculated blocking factor because a new schema is processed by DBSCHEMA.

One of the functions of the DBCONV program is to increase the length of the chain count field in non-standalone master datasets from 1 to 2 words (to be able to support chains with more than 64 K entries). This, in effect, causes a change in the media entry length but not a change in the blocking factor.

If a master dataset does not have enough residual space in its existing block to accommodate the extra word or words--namely, master sets which are blocked effectively--DBCONV increases its block size by up to four sectors. One sector, however, is usually sufficient.

There are two problems with the block size increases: first, the newly acquired sector is normally not being efficiently utilized, since it is just an overflow area of which as little as 1 word (of 128) could be used. The overall wasted disc space in a DBCONV-converted database can be quite substantial. These sets should be reblocked for greater disc and I/O efficiency.

The other problem is that the block size is increased for only some of the datasets. Remember, standalone masters, masters that have enough room for expansion, and detail datasets are not increased; also some sets may be increased by one sector while others up to four sectors. The result is that the block sizes of the datasets are inconsistent.

The drawback of inconsistent block sizes leads us into the topic of buffering.

Buffering

Whenever IMAGE/TurboIMAGE reads data from or writes data to a database, it moves the data into buffers which reside in an internal control block shared among all the datasets in the database.

The control block is sized to accommodate the largest block in the database, so if inconsistent block sizes are used in the database, data from some data sets are read into oversized buffers--a waste of valuable buffer space.

A buffer caching scheme not unlike disc caching is utilized in IMAGE/TurboIMAGE. For DBFINDs and DBGETs, the data are kept around in these buffers until all the available buffers are exhausted, at which time the dirty buffers are reused. In doing so, the buffers are simply overlaid with new data--since DBFIND and DBGET read but don't write, they do not update the buffers.

For DBPUTs, DBDELETEs, and DBUPDATEs, however, buffers are updated and must be written back to disc before they can be reused. IMAGE/TurboIMAGE is not comfortable leaving modified buffers lying around in its memory-resident control block, so it abandons its caching scheme and posts the buffers back to disc at the end of every DBPUT, DBGET, and DBDELETE intrinsic call. This is one of the reasons that IMAGE/TurboIMAGE is so reliable--not more than one intrinsic can be lost in the event of a system failure.

Output deferred and AUTODEFER

It is possible to force IMAGE/TurboIMAGE to keep these updated buffers from a DBPUT, DBDELETE, or DBUPDATE hanging around in its buffer control block rather than posting them back to disc until either they are needed for reuse or until the database is closed.

The resultant performance benefits of deferring posting can be substantial, but the risk factor is high because updated disc blocks from several intrinsics that can cover several datasets will be lost if a system failure occurs. In that event, you are certain to have both physical and logical database integrity problems.

The appropriate time to defer posting is for update tasks which can be redone so that if something goes wrong you can restore a backup copy of the database and rerun the process.

A substantial improvement in TurboIMAGE is the introduction of AUTODEFER, which allows output to be deferred for a database and which may be enabled and disabled via DBUTIL. Before Turbo/IMAGE, output deferred mode could be enabled for only one accessor.

Unfortunately, neither ILR nor rollback recovery can be used with AUTODEFER, and both must be disabled before AUTODEFER can be enabled.

Buffer management

In IMAGE, a fundamental throughput bottleneck was that not more than one intrinsic could execute at a time against a database since each intrinsic monopolized the buffer control block. This type of access is referred to as single-threaded, since only a single intrinsic can be processed at a time.

In TurboIMAGE, a new buffer management system was implemented, permitting intrinsics that require only one buffer (DBFIND, DBUPDATE, and some modes of DBGET) to access the buffer control block concurrently. This is called multi-threading. Intrinsics, however, which require more than one buffer (DBPUT, DBDELETE) are still single-threaded and maintain exclusive access of the buffer control block until completion.

To accomplish this, TurboIMAGE records dynamic information about each buffer's use in the buffer's 17-word header. When an intrinsic looks for a particular data block, TurboIMAGE searches the buffer control block to see if it contains that block. If it does not, TurboIMAGE selects a buffer for use based on a least-recently-used algorithm which considers, in this order, if the buffer is dirty, the possibility that the buffer will be needed again, and when the buffer was last used.

The advantages of TurboIMAGE's buffer management scheme is that it provides concurrent access for readers (users doing DBFINDs and DBGETs) and a smarter reuse of available buffers. Another benefit of TurboIMAGE is that more space is available for buffers due to a restructuring of the internal control blocks.

Buffer control block

In IMAGE, the control block which houses the buffers is called the DataBase Control Block (DBCB) and contains the Dataset Control Block (DSCB), a group of tables which reflect the database structure, and the Lock Area, which keeps track of the locks applied to the database. The DSCB occupies a fixed area determined by the size and structure of the database; the Lock Area expands and contracts dynamically between 128 and 4096 words as DBLOCKs and DBUNLOCKs are called. The residual space of about 28 K words is available for buffers.

In TurboIMAGE, because of the increased limits in database structure (199 sets instead of 99 and 1023 items instead of 255) the space required for the DSCB increased substantially and would not even fit into a single extra data segment. So up to five extra data segments may be required to house the renamed Database Global (DBG) control block, which now contains the Lock Area. The buffers were moved into a new control block--the Database Buffers (DBB) control block--which, after some overhead, has about 28 K words available for buffers.

Buffer allocation

Now, just because IMAGE/TurboIMAGE has a particular amount of space available for buffers does not mean that it uses this entire space; rather, it dynamically allocates buffers based on the complexity of the database and the number of users accessing it. The default buffer allocation strategy is calculated based on the average number of buffers required to do a DBPUT to the most path-ridden detail set. The formula is:

    4 * number of related automatic masters
  + 3 * number of related manual masters
  + 1

So for a database in which the most complex relationship is a detail set related to five masters, three of which are automatic masters, 19 buffers will be allocated ((3 * 4) + (3 * 2) + 1)

This algorithm sets the number of buffers for one or two users. For each additional pair of users, another buffer is allocated. The idea behind this tiered buffer allocation strategy is that only the required number of buffers is allocated at any time.

Default buffer allocation

IMAGE/TurboIMAGE's buffer allocation strategy seems to be sensible on the surface, since it minimizes memory usage by keeping the buffer control block as small as possible while still providing what it considers to be an adequate number of buffers. However, this default strategy is outdated and for several reasons should not be used.

First, the number of buffers allocated is based on single- threaded IMAGE--not multi-threaded TurboIMAGE--and does not take into account concurrent read access, which benefits from more buffers.

Second, the formula does not take into account other factors which may increase buffer demands such as ILR for DBPUTs and DBDELETEs.

Third, while dynamically sizing the buffer control block based on the number of accessors conserves memory, it cheats you out of potential buffers. This minimal memory conservation was far more important on 3000s of the early days, which supported less memory than your PC. Today, memory is cheap and plentiful.

Fourth, dynamically allocating buffers based on the number of accessors requires that the buffer control block be resized as each pair of users log on or off the database.

Lastly, a batch job looks like one user and therefore is, by default, assigned the minimum number of buffers. A batch job may, of course, be performing a lot of database access and should have lots of buffers.

Recommended buffer allocation

A better strategy is to have IMAGE/TurboIMAGE allocate as many buffers as possible all the time for any number of users. This means that the buffer control block is initially build to its full 32K size and does not change regardless of the number of users accessing it.

Having the maximum number of buffers assists both database readers and writers. DBFIND and DBGET benefit because there are more buffers to maintain the various data being accessed concurrently. This also increases the chances of having data blocks which are accessed by different intrinsics as part of one logical transaction still in the buffers (for example, a chain head from a DBFIND which is used for the subsequent DBGET, and then that entry used for the subsequent DBUPDATE).

DBPUT and DBDELETE benefit because the probability that enough buffers exist to complete an intrinsic call in one control block's worth of buffers is higher. If not, significantly more disc I/O is required, since when performing a DBPUT or DBDELETE IMAGE/TurboIMAGE first previews the transaction to make sure it can be completed.

For example, to DBPUT a new detail entry, IMAGE/TurboIMAGE has to check to make sure that free space exists in the detail set; that all related manual master entries exist; and that all related automatic master entries exist (and, if not, that free space exists to create them). To verify these things, IMAGE/TurboIMAGE may exhaust its buffers and require that they be refreshed with the rest of the data. And once it has verified that the intrinsic will complete, it has to again fill its buffers with the first set of data blocks, post them back to disc, and then do the same with the second set.

Naturally, this is not to say that every DBPUT will thrash the buffers this way. The number of buffers required by an intrinsic call varies not only on the intrinsic but the circumstances. For example, a DBPUT to a sorted chain will require a variable number of buffers based on how far up the chain needs to be read. A DBDELETE in a master will take a variable number of buffers based on whether the entry is a primary or secondary, the position on the secondary chain, and whether the chain is contained in a single block. A DBPUT to a detail set will require more buffers if the related automatic master entries do not exist, since they must be created.

Since the number of buffers may vary and exceed the defaults assigned by IMAGE/TurboIMAGE, it is better to have more buffers to absorb those intrinsic calls which require an inordinate number of buffers.

That explains why it is beneficial to have a lot of buffers, but why the same number of buffers regardless of the number of accessors? For each pair of users who log on or off the database, the buffer control block must be resized. To do so, IMAGE/TurboIMAGE locks the DBB, waits until all buffers are released by other processes, and the rebuilds it. This requires exclusive access to the control block and therefore impedes users waiting to process.

BUFFSPECS

Fortunately, there is an easy way of adjusting the number of buffers in IMAGE/TurboIMAGE. Just specify them in DBUTIL, for example:

```
      >>SET base BUFFSPECS = 30(1/200)
```

This command allocates 30 buffers for anywhere between 1 and 200 database accessors. This causes a consistent number of buffers to be allocated with no dynamic sizing of the control block, and allocates more buffers than IMAGE/TurboIMAGE's default.

Of course, the obvious question here is "how to you determine how many buffers to allocate?"

For TurboIMAGE, you don't need to make this determination. You can simply set an impossibly high number of buffers (for example, 255) and TurboIMAGE will allocate the maximum number of buffers possible.

Now, don't try this for IMAGE because specifying more buffers than can fit in the available space can cause a buffer supply crisis and resulting problems. The reason is that the both the buffers and the Lock Area are contained in the same control block (DBCB) and both areas are dynamic. The Lock Area expands "downward" while the buffer area expands "upward," and if too many buffers are allocated and the Lock Area needs to expand, the two collide. This problem does not exist in TurboIMAGE because the Lock Area is contained in a different control block.

For IMAGE, it is necessary to calculate the number of buffers. To do this, you first need to know how big the buffers are.

Buffer length

Just as the number of entries that may fit into a block is determined by the entry length, the number of buffers that can fit into the buffer control block is determined by the buffer length.

The buffer length is calculated based on the largest block size in the database, since the buffers are shared among all the datasets with any block being able to be read into any buffer. The method is to take the block length (BLK LGTH from DBSCHEMA's dataset summary table, which represents the blocking factor times the media record length plus the bit map) and add nine words for IMAGE or 17 words for TurboIMAGE (for the buffer header). Unlike the block size, the buffer length is not rounded up to the nearest 128-word multiple.

How many buffers?

Now that you have calculated the length of the buffer, how many of them can you have? To determine this, you need to figure out how much space is available for buffers. The formulas are different for IMAGE and Turbo/IMAGE.

For IMAGE, you subtract the fixed space needed for tables and the maximum possible Lock Area from 32 K words, and this yields the available buffer space.

Let's try it using the same database from the TurboIMAGE manual. The information in the DBSCHEMA output below the dataset summary table is needed for this calculation; to calculate the available buffer space for this database:

```
ITEM NAME COUNT: 23         DATA SET COUNT:  6
ROOT LENGTH:   1176  BUFFER LENGTH: 505  TRAILER LENGTH: 256
```

goes like this:

32767 words	(32K, maximum DBCB size)
- 150 words	(fixed overhead)
- 1176 words	(ROOT LENGTH)
- 23 words	(ITEM NAME COUNT)
- 6 words	(DATA SET COUNT)
- 256 words	(TRAILER LENGTH)
- 4096 words	(maximum Lock Area)

The result is that 27,060 words are available for buffers. Since the buffer length for this database is 521 words (512 + 9), 51 buffers as most can be allocated (27,060 / 521).

This formula changes in TurboIMAGE. The calculation for the same database in TurboIMAGE is:

32767 words	(32K, maximum DBB size)
- 4000 words	(fixed overhead)
- 156 words	(26 words * DATA SET COUNT)

which results in 28,611 words available for buffers. In TurboIMAGE, the buffer length increases to 529 words (512 + 17), so 54 buffers may be allocated (slightly more than in IMAGE).

Determining block size

Now that we've reviewed what blocking and buffering are all about, how they work, and how to adjust them, let's focus on the fundamental element in all of this: block size.

The block size ultimately determines disc storage efficiency, the amount of data transferred from disc to memory, the size of the buffers, and the available buffer space.

Basically, small blocks mean lower blocking factors and more buffers; large blocks mean higher blocking factors and fewer buffers. Choosing a good block size is fundamental to the performance of your database.

The TurboIMAGE manual lists some guidelines in selecting a BLOCKMAX (and thereby block sizes). It says you should consider efficient space utilization, minimum disc accesses, and program execution time which can be affected by size of the DBB. It recommends using larger blocks if applications run in batch when system resources are plentiful and smaller blocks if applications are large and run concurrently with many online users. This is because the resulting large DBG and DBB may cause applications to run more slowly since a large area of memory is required, and this may not be necessary for small applications.

The concerns about program execution time are generally not significant, since most systems have enough memory to comfortably support large control blocks. As far as efficient disc storage goes, larger block sizes generally store data more efficiently (although DBSCHEMA will ignore the BLOCKMAX to save some disc space).

The concern about minimizing disc access require a closer look.

Disc I/O bottleneck

The most common basic performance problem in IMAGE/TurboIMAGE and other systems on the HP3000 and other computers is that data cannot be moved to and from disc fast enough: the system is ready to process data but it must wait on the disc drives for the data to be retrieved, so a fundamental bottleneck in the system is disc I/O.

It is always beneficial to reduce the amount of disc I/O, and one factor external to IMAGE/TurboIMAGE which must be considered here is disc caching, both in memory (system disc caching) or on the disc drive itself (controller caching).

Disc caching

The concepts of disc caching are quite simple: with each disc I/O, you read more data then you need right now with the expectation that you might need it soon--after all, as long as you're going to spend resources on a disc I/O, you might as well get as much data as you can with about the same overhead. Also, data that is in demand remains in cache while unpopular data is flushed to make room for data from subsequent I/Os.

For system disc caching, the amount of data retrieved is determined by fetch quantums set via the :CACHECONTROL command. For IMAGE/TurboIMAGE, only the random fetch (not the sequential fetch) quantum is applicable--even for sequential access of a dataset. The fetch quantum determines, for each disc I/O performed against a given location, how many sectors following that address are also pulled from disc in the same I/O.

Disc caching acts as a cushion for such deficiencies as low blocking factors and poor data locality. A cached I/O against a dataset with a block size of 512 words and a blocking factor of 3, and with the random fetch quantum set for 96 (the maximum), will transfer 72 records from disc to memory. Remember, that the data still has to be read into the buffer control block a block at a time, but this is many times faster than a disc I/O.

There are situations in which disc caching does not help or is a hindrance to database access. For instance, you may be accessing data which in the dataset physically precedes the data read previously, which is a common occurrence when reading backward up a chain (DBGET mode-6) or when accessing entries whose locations were determined by reusing space on the delete chain. Since caching reads forward only, useless data is swapped into cache and then must be searched through before going back to disc. The result is increased overhead from the memory manager.

Large blocks versus small blocks

In the above scenario, bigger blocks are preferable. Serial reads especially benefit from big blocks because many qualifying entries are read with each disc I/O and a lot of in-demand data are read into the buffers at once. The same is true for chained reads of data with good data locality.

Small blocks mean low blocking factors, which suggest increased disc activity. If less entries are contained in each block, more blocks have to be read from disc to get all the required data. The benefit of disc caching here, however, should not be minimized. Small blocks also mean more buffers, which benefits concurrent read access and cushions DBPUTs and DBDELETEs.

Regardless of the block size you select, calculate the number of buffers required for a DBPUT to the most complex detail set and make sure at least that number of buffers can be allocated. If not, buffer thrashing will result and you should reduce the block size.

Summary

This summary of the recommendations explained in this article. Use them as guidelines--not rules--in optimizing blocking and buffering in your databases:

- override DBSCHEMA's calculated blocking factor when it results in a block size less than the BLOCKMAX

- recalculate the blocking factors for datasets that are initially created with very low capacities and whose capacities are increased

- avoid designing datasets with very long records, since they result in very low blocking factors

- avoid designing datasets with records which cannot be effectively blocked

- reblock DBCONV-expanded TurboIMAGE master datasets to be more disc and I/O efficient

- enforce a consistent block size for all datasets to maximize available buffer space

- use output deferred mode (AUTODEFER) for write-intensive tasks which can be rerun in case of failure

- override buffer allocation defaults to allocate the maximum number of buffers to improve throughput. For IMAGE, calculate the maximum; for TurboIMAGE, assign an unattainable value and the maximum allowable will be assigned

- override the default allocation of buffers based on the number of accessors; assign a fixed number of buffers for any number of users to prevent the buffer control block from being rebuilt

- determine an efficient BLOCKMAX and make sure that it does not limit the number of buffers to the point that they thrash

EDI ENHANCES JIT OPERATIONS

by: Charles S. Townsend
President, Birmingham Computer Group, Inc.

Good morning and welcome to my presentation on "Electronic Data Interchange". My presentation today will be divided into two parts. First, I will discuss what the nature of Electronic Data Interchange (EDI) is, how the transactions are formed, what the history of Electronic Data Interchange has been, and what the future offers for Electronic Data Interchange.

As a member of the Automotive Industry Action Group (AIAG) and the ANSI X12 Standards setting committees, I have had a significant amount of experience in the formulation and development of Electronic Data Interchange Standards that are being proposed for national usage.

During this presentation, I will discuss how these Standards are formed, why the Standards are formed in the way that they are, and why the use of these Standards would benefit the users.

In the second part of the presentation we will talk about the nature of the Standards, what characteristics the Standards have, and what these characteristics mean to the eventual users of the Standards. For example, characteristics such as uniformity, flexibility, machine readability, instantaneous communications, efficient transactions and so forth, --- all have dollar and cents benefits to those organizations which use Standards. Because of the dollar and cents benefits, EDI is becoming widely used and is projected to become much more widely used in the near future.

It is important to all of us, who are providing computer services or tailoring computer software to business activities, to understand the role of Electronic Data Interchange and to understand where the benefits from Electronic Data Interchange may be garnered. This is the topic of this presentation.

Electronic Data Interchange is a means of sending transactions from one computer to another. This is more than simply networking, where we may be sending transactions between two computers using the same operating systems and using the same software. In Electronic Data Interchange, we are sending transactions between all kinds of computers (different types), all different types of operating systems, all different types of software that are in use in business today. Often this includes the use of a public network.

The example on the slide shows that you can send Electronic Data to your trading partner. With a buyer, in one case, you can send a purchase order; and, he may send back an invoice to you, when he has delivered the material which you have ordered. With a customer, you may receive a purchase order and send back an invoice to your customer. So, the transactions from your point of view may be inbound or outbound transactions. You'll see on the graphic that is depicted on the slide, that we are assuming direct computer to computer hook-ups. That is, my computer has an auto-dial modem; your computer has an auto-answer modem; and, when I dial your telephone number it rings your computer; and, the two computers talk to each other. I send you transactions and you send me transactions. This is the way that Electronic Data Interchange is being done widely today.

The next slide we see an alternative means of Electronic Data Interchange where I do not talk directly to my suppliers or buyers. Instead, I send (all of the documents I wish for them to receive) to a third party network.

This third party network is like the United States Post Office. They have electronic mailboxes and I may send as many purchase orders in one transmission to the third-party network as I wish. I may send five purchase orders to five different suppliers. At the same time I can send invoices to my customers, I can send requests for quotes to my suppliers and so forth.

When the transactions reach the third party network they will be broken down and put into the various supplier's and customer's mailboxes; and, later when those suppliers and customers inquire against their mailboxes, they will see by transactions and they will be able to respond to them much the way they would if they had arrived in the morning mail --- except much sooner, the same day that I sent them. By the same token, my customers and buyers will send transactions back through the third party network to me; and, I have a mailbox on the third party network. So if some of my customers send me purchase orders and some of my vendors send me quotation responses, they will all wind up in my mailbox on the third party network. And, I will periodically be able to dial up the third party network and get those transactions out of the mailbox. Can you see how this is starting to resemble the post office?

In this slide we are showing the interaction of several different industries. As you can see, by the dinosaur in the lower left-hand corner, that we don't expect industries or corporations, which remain out of the Electronic Data Interchange game indefinitely, to prosper. We expect that they will be bypassed because they don't have the capabilities to conduct business in the way that business will be conducted in the future --- with electronic trans actions. This also shows that any kind of transactions, electronic or paper generated transactions, are not limited to a single industry or do not follow industry lines. Whereas we often say the automobile industry uses material releases and advance shipping notices, those transactions are not solely the property of the automobile industry. The aviation industry uses material releases. All industries use shipping notices when shipments have been made. Sometimes they are called shipping manifests; and, these transactions can cross industry lines. A single corporation may be selling in the electrical industry, the aerospace industry, the photographic industry, the tobacco industry, the retail industry and the automobile industry --- all at once, depending on what it is they make and what they do. Fabric industries, for instance, may make sales to all of those industries. So these transactions the Fabric Industry would be receiving and sending back out are not limited by industry; and, neither do we limit the Electronic Data Interchange transactions to a single industry. These are, in fact all industry transactions.

The history of Electronic Data Interchange started with various industry groups. For instance, the transportation industry had a need to transmit and receive bill of ladings and shipment information. The volumes that the rail industry had in terms of high volume, low value transactions were so great that they had to get them done efficiently. They couldn't afford the $27 or $34 per transaction that was going on in the paper generated arena. So, TDCC came up with some Standards and started employing those for electronic data communications of freight bills. UCS is sponsored by the retail industry; and, the retail industry, of course, has many small suppliers of garments, fabrics, tools, toys, items that are sold in retail stores. So, they came up with a Standard especially supported by the grocery industry. At the same time, the aerospace industry came up with Specification 2000 and the Government started a Standard they refer to as CALS. And European industries started using a standard which they refer to as ODETTE. The warehousing industry started sending purchase orders and invoices in a Standard which they called WINS; and, the Automotive Industry Action Group started a standard for themselves.

Electronic Data Interchange is traced back earlier in history than you may think. Aerospace began in about 1958 to use an early form of what is now known as SPEC 2000 as a way for airplane manufacturers to sell parts to the airlines which maintain the airplanes. This standard has grown over the past 30 years and is now known as Specification 2000. It is widely used among those people who supply components and sub-components in the Aerospace Industry or who order these components.

The Auto Industry began in the early, mid-sixties to begin to order parts for their assembly lines through what they call "material releases". In the early days each division in the automobile industry was pretty independent and provided a material release format that each would send out to all of their suppliers. In each of these material release formats were usually fixed, eighty column records. Chrysler for some reason uses 146 column record. But in any event, they are sending out these fixed records saying in Record A we'll tell you what the part number is, and we'll tell you how we want it shipped, and in Record B we'll tell you the date and quantities to ship in first week, and in Record C we'll tell you the dates and the quantities to ship in the second week. And these were pretty effective, because computers were simple enough in those days. You couldn't afford to send a lot of data back and forth. This worked; they could talk to their big suppliers, the steel suppliers, the glass suppliers and so forth. This was pretty effective; but, one of the problems with these kind of fixed record standards is that you have to allow a field large enough for all your numbers. So if you can order a million parts, you must have a seven position field for "quantity". If the next guy coming along only orders 15 parts, he uses two positions of the seven position field and there are five zeros taking up space and taking up telephone time, considered a valuable resource. The Auto Industry Standards were somewhat inefficient in that they were transmitting a lot of spaces and a lot of zeros along with the real live data that they were transmitting. Each of the auto companies had these fixed records. General Motors had theirs; Ford had theirs, even by division they had fixed records for different purposes; and, then on the outbound side, they would expect the suppliers, (when they shipped some material for the assembly line) to send an advance shipping notice, so that the Company would be forewarned that this material was on the way and they could prepare the shipping dock for receipt of shipment.

So these Standards grew up for various reasons and, of course, ANSI is the American National Standards Institute for setting such standards and their X12 Committee has been designated as the Committee to set Standards for Electronic Data Transaction Sets. X12 Committee does not set Standards for the protocol of the modems that will be talking to each other. That's done by X25 and they do not set Standards for log-on records and envelopes that will be around the records. That's done by X400. What X12 does is specify what the transactions will have inside and what kind of data there will be what form the data will be in, how the data will arrive, how the users may interpret the data. Anyone who wishes to, may become a member of the ANSI Organization. Whether or not you are a member, you may also attend the Quarterly Standard Setting Meetings, which are held May, August, November and February, a week at a time. At these meetings, sub-committees are formed. There's a Goverment Task Force, a Materials Management Task Force, a Purchasing Task Force, a Financial Transactions Task Force and then there are some other groups that control the syntax, languages and so on. You may join any of these task forces or sub-committees. When you attend the Standard Setting Meetings, you will have your voice heard. Standards are set by consensus of all those that are participating in the Standards Setting arena. If you become a Committee Member in the Standards, you may receive voting packets and until you vote affirmative on a transaction set, it cannot be accepted.

The people who provide the transaction sets for voting will also provide responses to all of the NO votes. When you do vote NO on a transaction set you are expected to give a reason why you voted NO.

In any event, about four years ago, ANSI got more serious about setting electronic data standards and they have since that time defined several transaction sets such as request for quote, quotation response, purchase order, purchase order acknowledgement, purchase order change request, purchase order change acknowledgement, invoice, payment application, material release, just-in-time delivery schedule, advance shipping notice, stock status, inquire stock status response and several others. All of these transactions can be sent back and forth between what we call "trading partners".

ANSI only controls National Standards and there is another Standard that is being developed called the EDIFACT Standard. This Standard has been proposed in the United Nations and accepted as a world-wide Standard; and it, like ANSI, is under development by committees around the world which are proposing that a purchase order should look like this and so on and so forth. The syntax at this point between ANSI and EDIFACT is not the same. But, there are some movements afloat especially, within ANSI, to have changes made in either the ANSI, or the EDIFACT, or both Standards in order to bring the syntax differences together.

Since most of the Standards were originally patterned after the work that was done by TDCC, the ANSI National Standards, the AIAG Standards and the UCS Standards all look very much alike; and, they are very much variable record. The advantage of being variable record is that the data elements do not have to be in the fixed length but can be of whatever length is necessary to transmit the needed information. So, in my earlier example, if I am to order 1,553,723 items I need seven positions in quanitity volume to transmit that order number. If you are ordering 15, you only need two positions. I send seven positions you send two positions. How does the user, who gets the information on the other end, know that this is the only number of positions that I'm sending? We'll go into that in a minute and we will see what the Standards look like. The reason we are able to do that is that we are sending variable standards. The theory is that we will save 20 percent of the data bytes, which are currently being sent in fixed records standards, by sending them in variable record standards.

As you can see by this slide, this is a sample slide of some of the industries which are getting extremely serious about using Electronic Data Interchange with their members. They are going to their local associations, national associations. They're adopting the ANSI X12 Standards by and large internally; they're learning how to use the ANSI X12 Standards and starting to demand that their trading partners begin to use the ANSI X12 Standards. The Automobile Industry, of course, has been demanding this for quite some time. The Chemical Industry recently has been very strong in this in demanding similar usage. The Retail Industry now is beginning to start up and demand some usage of these transaction sets. There is an indication that, while there are approximately 5,000 or 10,000 companies using EDI today, there is a prospect of roughly a million companies that will begin to use EDI. We expect that most of them will probably have Electronic Data Interchange installed in their organizations like a FAX machine in the near future.

What does an ANSI X12 Electronic Data Transaction look like? This slide shows in a paper transaction, which you are all probably familiar with. This one happens to be an invoice. On an invoice we have basically two pieces of information. We have header information and we have what we call detail or line item information. Since line item information can repeat with line 1, line 2, and line 3, we say that the line item information can loop. That is we can have that information once for line 1; we can have some more information of the same type for line 2 and so forth.

On this slide we see the data comparison between data that is maintained in electronic form and data that is maintained in a paper form. This is a small sample of the data that you might see in Electronic Data Interchange transaction. This is what is known as a data segment. Electronic Data Interchanges are broken down into three levels.

At the higher level you would have what we call transaction sets. A transaction set would be an entire invoice with all of its header information, all of its line information, all of its totaling information. An invoice would be a transaction set. Within a transaction set we have what we call data segments. Data segments are discreet strings of data that have a related meaning. For instance, we have some data segment information in the N1 Record. The N1 is the Name Record or the Name and Address Block, N1, N2, N3 and N4. So we have name and address information in those four data segments. We may have, for instance, a code specifying what the address is. For instance, an ST would be "Ship To" address. Then we may have another code specifying what kind of identification number we are using in this "ship to" segment. For example an 01 code indicates that we are using a DUNS number. Then the following data element is the DUNS number itself. So now if three data elements have ST, (which is "ship to") or SF, (which is "ship from") and I have 01 which means I am using a DUNS number, then I have the DUNS number itself and finally I may have the Company name itself printed out there. I do not need to have the Company name. All of this information strung together is a data segment, its the name segment. Within the data segments we have data elements. Data elements are the little pieces of information that make up the data segments. So the ST, which indicates what kind of address this is, is a data element. The 01 is another data element. The DUNS number itself is the third data element; and, the name of the Company is the fourth data element. So, in the example I have given, we have four data elements in an N1 segment.

These segments are transformed into something that looks like what we have on this slide. You will notice that the data has been squashed together and does not have any leading zeros and does not have any unnecessary spaces in it. You will also notice that there are funny little asteriks in between each data element. These are called data element "separators". We need data element separators because we do not know the length that each data element will be. Its going to be various lengths. We use the data element separator to tell us that we are at the end of that particular data element. We know from the Standards that the first data element tells us the segment identifier so that we know what segment we are dealing with, then we have an asterisk and the next one is a data element and tells us some data; that's the first unit of data. Asterisks are typically not used in this application but are only used for graphic purposes. Usually, some characters, sometimes unprintable, are used. Tilde is a common character, tilde is frequently used to indicate the end of a data element. And then you will notice at the end of the line is the NL which we call the data segment terminator. The NL stands for the mainframe configuration new line but that again can be any unprintable character which terminates the data segment. So within the Standards there aren't any standard syntax elements for data element separator and the data segment terminator. They may be determined by the sender of the message at the time that he sends the message. In any event this is what an ANSI X12 transaction looks like.

In order to send transactions through third party network, so that it can be found, all of this transaction information has to be enveloped. We will envelope similar transaction sets within an inside envelope called a functional group envelope. We can gather together, say if we are sending invoices, many invoices (and each invoice is a transaction set). But, all of the invoices taken as a group can be a functional group envelope of invoice transactions; but, within a functional group, all of the transactions must be the same. Then we may take many functional groups together to create an interchange envelope which is a complete transmission. And, that transmission like an envelope can be sent out. So I can send a transmission to Company A,B, C and I may have both purchase orders, purchase order change requests, and invoices. I can send those out to the third party network and the network can see all of those transactions because of the envelope structure and put them in A, B, and C mailboxes.

During the same transmission, I can also send other transactions to one of my other vendors or one of my other customers and put them in their mailbox. They will also be enveloped with set envelopes around them and then group envelopes around those and finally the transaction envelope around that. Again the enveloping is a Standard.

All of this Electronic Data Interchange business is very interesting. It keeps all of us on the Standard Setting Committees kind of busy. But the real issue for the user of Electronic Data Interchange is how can I benefit from communicating these transactions faster. Certainly the benefit is there if you don't have to wait for the mailman to come in the morning with your purchase orders, so that you can fill your purchase orders, since he may have been holding on to them for two or three days. Certainly, you're going to pick up a day or two days of mail delivery time. That's nice, but that's not what you're really looking for. In addition to that you want to be able to get that data integrated into your operational systems quickly and get the operational systems moving with that data in them. How do you do that? That is the essence of Electronic Data Interchange. It's not just bringing the data in the door and setting it down in the corner. It's getting the data in the door, internalizing the data in the operational system, and then using the data to make business decisions or process business transactions and get your business out of the way quicker, faster and more efficiently.

This slide shows some of the internalization that needs to take place when data transactions are received or sent from customers who may send you purchase orders and receive invoices back from you. You may send them shippers as well and you can send them electronic mail. On the vendor side, which is the lower box, you can see that the transactions are the same but they are just reversed because now you are the customer and the vendor is the vendor; and he will be sending you the invoices and the shippers and you will be sending him the purchase orders and the purchase order change requests; and, he'll be sending you the acknowledgements. But in every case, whether you're talking to your customers or your vendors, you wind up moving the data into your applications. And, this dotted box on the right hand side of the screen shows the various applications which might be receiving some of this data. You might move some of the material release data into your scheduling; you might move the purchase order into your order entry system; you might move the invoicing data into your accounts payable system and so forth. We will see in a minute the savings that can be made if you can move these items using EDI. That concludes the discussion of what is Electronic Data Interchange and what are the ANSI X12 Standards. Do I have any questions to this point. We'll allow 10-12 Minutes for questions.

Moving on to part II...

The goals of part two of the discussion are to show the attendees seven characteristics of the ANSI X12 Standards and how each one of these characterisitics can provide them with an internal savings or efficiency that should improve their competiveness, if properly implemented within their business situation. And, we have some real life examples that depict this actually happening.

This first slide shows that the Standards promote a uniformity among industries and a uniformity among businesses. In the early days as was mentioned, the Aerospace Industry went off and did their own thing. The Auto Industry was even worse than that, they didn't go off and do their own thing. Rather, each automotive manufacturer went off and did his own thing. There wasn't any uniformity. And even today, with ANSI X12 Standards on the horizon and some available for use, the Auto Industry is still clinging in some quarters, and will continue to do so for several years, to their fixed record standards. And where they have the fixed record standards, people who provide software for that environment need to provide a system for Ford, another system for GM, another system for Chrysler and so on and so forth. When the ANSI X12 are adopted, Ford, GM and Chrysler will still say we don't want to use the same data elements as the other guy uses. Our business is different somehow. However, they will use the same construction of the data. So now we can simplify our software a little bit by saying we don't know what kind of information Chrysler is going to send us but we do know that, if it is name address information its going to come in an N1 segment. If it's part number information it's going to come in a segment that begins with an LIN. If its quantity, if its forecasting information, its going to come in an FST segment. So we have introduced a great deal of uniformity. Uniformity is not only introduced throughtout the Automotive Industry and beneficial there but this uniformity is continuing to be introduced throughout all of the industries. The Department of Defense for the Federal Government has said that they will use CALS as an internal structure to control their data processing development. But when they go outside of the Defense Department into the public industry to buy material for the Defense Department, they will in the future begin to use ANSI X12 as their communication standard, recognizing that the industry in general is using that standard; and, they will coordinate with that.

At first, in the early days, we may have had a trading partner who had a great big HP computer and they may have said, well we have an inventory system and you can inquire into our inventory system here. Take an HP terminal; set it up in your office; buy a modem and you can hook that HP terminal into our computer and inquire into our data system. Well that's fine except for one thing, this isn't my only customer. I may have 100 such customers and they may all want me to get their own brand of terminal. They may all have their own inventory stock status system, which I have to learn how each one of them works; and, then if I want to do some stock status order entry with any one of them, I have to go into their systems. This was fine in the early days when their wasn't anything better; but, this causes me to have to coordination problems with many different systems in my office.

By the creation of the ANSI X12 Standards, and across the Industry Implementation Guideline, we now have a single way in which data can be transmitted. It allows us to buy a single computer system which will be able to send and receive those transactions to all companies that we will be communicating with in the future, regardless of what industry they may be in. That is the benefit of uniformity. I do not need to learn different packages or purchase a different package for each different trading partner that I have.

The second characteristic is flexibility. When you go with a communications standard that has fixed formats and it says that the quantity field is seven characters long and I want to order three billion quantities I can't do it in one quantity field, I don't have enough spaces to do that. ANSI X12 standards are more flexible than that. They say the quantity field can be up to 16 characters long but you can only use as many characters as you need. So, if you only need two or three characters, you just use those. The Standards are flexible and we don't have to be as concerned with the length size of the fields. The standards do specify a minimum and maximum length for each standard, so that the fields can't go on and on forever. They all have a specific ending length. The Federal Government came up with an interesting problem. When they send out a Purchase Order or a Request for a Quote, they need to tell their suppliers about 2,500 different clauses that the Federal Government and Congress has imposed on their purchasing such as: you need to buy products that are made in America, that you need to buy products from companies that have a minority representation, that you need to buy products that meet certain specifications. There are over 10,000 such clauses in the purchasing practices of the US Government; and, it is not unusual to see 1,500 or 2,000 of these clauses applied to a single purchase order. Well, because of the flexibility of the standard, they say you may have clauses, you may have reference numbers in here, and each clause is assigned a reference number. If the standard says we don't care how many reference numbers you put in there, then you can put 2,000 reference numbers in there for that single purchase order line item.

Think about describing a telephone pole for example. What is a telephone pole? Is it a stick of wood? No, it's basically a tree trunk that is oh so big around and is so tall. We have to specify how tall that's going to be, we have to specify what kind of wood its going to be made out of; what kind of decay inhibiting solution the wood is going to be soaked in for how long, how much weight the wood is going to gain, where the holes are going to be drilled, how many cross members are going to be put on, how many little nitches (for the guys to climb up the pole), where those nitches are going to be located. It can take you five or six pages to describe a telephone pole.

The ANSI X12 Standards are not limited to their descriptive length so they can accommodate comments and more comments on a purchase order of that length so they are flexible. Because the standards are flexible, they can cover all the idiosyncrasies of the various industries and yet can be tailored to each trading associate. So the flexibility allows the Standards to be used across all industries. If the Standards were less flexible than many, many industries, perhaps all industries would say: Hey, we can't use the Standards; you'll just have to get some other Standards, because it doesn't match our industry. The ANSI X12 Standards are very flexible.

The next characteristic of the Standards is machine readability. The real benefit of Electronic Data Interchange is that a transaction is generated in Computer A and without having been printed out on hard copy is sent to Computer B and is internalized in Computer B, (Let's say into the Order Entry System) without anybody having to sit down and type in all the detail of the "ship to" name and the "ship to" address and making errors perhaps and putting in wrong quantities, sometimes inverting numbers, etc.. When Electronic Data Interchange is working efficiently, those things don't happen. Those operator keypunch errors do not get introduced into the transaction set on the other end.

It has been estimated that 70% of the material that is inputted into one computer and outputted by another computer; and, about 70% of the information that is outputted by a computer is inputted by another computer. So how much time are we wasting in our business if we continue to dump these long reports and all these forms, print out all this hard copy, run over to another computer, give it to an operator, have the operator sit down and batch this data back into the second computer when we have an alternative. We have a way of sending that 70% of the data from this computer to that computer in a matter of seconds or minutes and having it already in there without typographical errors. It allows us not only to save on the existing transactions but it allows us to dream up more complex transactions which we would never attempt with manual entry; but, now we can attempt the transactions because we have the computer doing the entry and it takes only seconds. We would never have attempted to put the "ship to" addresses in for all of the Chevrolet repair facilities in the United States, all of the little bump shops that could buy parts to repair cars, because there are tens of thousands of them. And all the time that the operators are entering these addresses they could be entering them incorrectly and the parts may be shipped to the wrong spot. With Electronic Data Interchange, we don't enter the names and addresses. The dealer enters his own name and address. He enters the name and address, we get electronically into our own computer. We don't have to spend time putting it in and we can send the part to him.

Machine readability allows 70% of computer posting quickly and allows for more data caputure. The benefits of this are reduced clerical effort and reduced material handling. How does it reduce material handling you ask. Let's take this example. In the old days General Motors would come to me and they would say I want to buy five million steering shafts from you. I'm going to install some in Buicks and some in Oldsmobiles. So I would send them five million shafts and they would take them to some warehouse located half way between Lansing and Flint (where Oldsmobiles and Buicks are made); and, as they needed steering shafts in Lansing they would be sorted out of the warehouse and sent to Lansing. And, as they needed steering shafts in Flint, they'd be sorted out of the warehouse and sent to Flint. With the advent of Electronic Data Interchange, Buick and Oldsmobile begin to order their own steering shafts. Now I am not sending the shafts to General Motors (one location) I am sending some shafts to Buick and some shafts to Oldsmobile. This means they do not need that interim warehouse. The shafts don't stop there, we don't have a truck line taking them to the intermediate warehouse and leaving them there, forklift trucks putting them away and then other forklift trucks taking them down and sending them out. With Electronic Data Interchange, I now can note the specific factory which I want the steering shafts sent to. I may not have been able to know that with non electronic capabilities in the past. I have saved a truck ride load and unload, a lift here a lift there. I've saved a lot of material handling costs. Now with the Just-in-Time environment that the Auto Industry is attempting to adhere to, we save even more time. Now what we are trying to do is send the material not only to that factory but to that specific spot on the factory floor, where the material is going to be used to be built on to the car. Even within the manufacturing factory we no longer have forklifts storing the material, picking it up and moving it around. So we have reduced material handling.

The *fourth characteristic* of Electronic Data Interchange is Instantaneous Response. We now are looking at an industry where, as the cars come out of the paint shop, they're assigned a VIN number the VIN number is hourly broadcast to the supplier who put the seats, dashboards, headliners and fenders on the cars. These suppliers can either pick from inventory the specific sequence that the material is going to be sent to the assembly line for building into that car. Because they know the sequence that the cars are going to be built in, they know that they send a blue cloth seat next and then they send a red leather seat and a brown cloth seat, and a blue vinyl seat. They go on to the truck, come off of the truck in that sequence, and go onto the assembly line into the cars in the sequence that the supplier has packed them. This is called "line sequencing". Huge savings in inventory reduction. The automobile companies no longer store four or five hours of seats on site, try to pick the seats out of their storage and put them into the sequence for the line. The truck shows up at the loading dock, the seats are taken right off of the truck and put right into the cars without stopping. With the ANSI X12 Standards, the information is sent in efficient transmitted transactions which are batched, unnecessary data is removed. That is, the zeros and the spaces are dropped and the data speeds are increased so we send more data, less air time. And, we're more efficient.

The sixth characteristic of Electronic Data Interchange is Organized. Once we start to try to communicate with ten or 11 trading partners, it gets complicated. We're trying to dial the phone up; we're trying to send these transactions to these guys; those transactions to those guys; and, it gets very confusing. With mailboxing and third party networks, we can make a single call and get economies of scale and send all of our transactions up to the third party network and have them mailboxed for us (of course there is a fee for that). Thereby, we reduce our electronic equipment and our operational complexity. We actually have companies in the Auto Industry, that have 50 to 60 modems standing there answering the phones all day long. Thousands and thousands of calls coming in that can be reduced.

Finally, the seventh characteristic is Affordability. With the Standards being uniform and the state of the art of the technology progressing rapidly we now have an increase in the availability of systems which support the ANSI X12 Standard. Because we have more systems which are available we have better communication systems. The price of the hardware and even some of the pricing of the software is coming down and you can get more and better data now at a lower cost than ever before. EDI systems are very affordable at this time.

In summary, Electronic Data Interchange is a new opportunity for a business to gain benefits in both customer service and productivity. Once the tangible business benefits of EDI are understood, the use and growth of EDI throughout all industries will be phenomenal.

Rolf Schleicher
Harksheider Strasse 29
D-2000 Hamburg 65
West Germany
Phone (private): (49 40) 602 48 25
Company: Deutsche BP AG, Hamburg
Hamburg, June 29. 1988

Low-cost, high efficiency with integrated PCs - Don't reinvent the wheel

Abstract

Only a few EDP-shop managers have already realised the power of Personal Computers in HP3000 environments. Others still ignore PC software with easy-to-use, low-cost user-interfaces. PC's are often simply used as dumb terminals.

However, with today's powerful terminal-emulators you have the tool to integrate the world of PC-programs with existing HP3000 applications.

PC-hardware and software is more economical than HP3000 software and often much more user-friendly.

HP3000-programs can control your PCs and use ready-made PC programs to carry out HP3000 work overnight.
PC-programs can use HP3000 data without file transfer (direct access).

With your HP3000 programs, you can even do things that you never dreamed of. For example: showing pictures, dialing your phone, using on-screen calculators, using keybord macros, switching between up to four running HP-online sessions with one keystroke.
Examples of powerful PC programs demonstrate that you can greatly enhance your existing HP3000 applications at a low cost without any program modification. Some of these PC programs do the work of $ 5000,- HP programs at bargain-prices of $ 50,- or less.

Introduction

More and more users have become familiar with PCs and apperiate the many powerful programs which are available.
They are beginning to ask the HP computer-shop why data processing with these systems is often not very user-oriented and why the generation of an application is so time-consuming and expensive.

Since firms fully utilize the computing capacity of their HP 3000 and since there is a growing backlog in most EDP departments, the end users are beginning to solve the problems which they have with their PCs themselves.

If this development is not properly channelled, great integration problems will result with the existing host applications due to incorrect, undocumented end user applications.

Therefore, the integration of PCs must be planned and regulated by the EDP department.

The EDP department can often assist the users more quickly, efficiently and cost-effectively by methodically including the PCs and the many efficient PC programs available than by buying or developing HP programs.

What is integration?

I understand "integration" to mean the inclusion of the PCs and the integration of PC programs into HP 3000 data processing.

Points of emphasis:

1. Terminal emulation
2. File transfer
3. Improving HP applications by using PC utilities
4. Resource sharing
5. Distributed data processing
6. End user support during planning and creation of applications
7. Problems with integration.

You do not have to wait for announced PC integration programs such as "New Wave" or 'PS/2'-systems with 5 MB storage and large hard disks. In many cases, you can work with the small PCs available and proven PC programs. Even expensive HP 3000 applications for example 'HP Access', 'Business Report Writer' and word processing programs can be replaced by low-cost PC programs.

The most important PC tool for the integration of the HP 3000 and PCs is a good terminal emulator. Our company decided to use REFLECTION 3+ from Walker Richer & Quinn. The following text is based on the features offered by this program.

1. Terminal emulation

PCs are generally only used for local processing (e. g. word processing, spreadsheets) and in addition as a terminal for the HP 3000.
The possibilities of terminal emulation are often not fully explored.
Here are a few features of terminal emulation which alone justify the use of PCs:

- 'Type Ahead' extremely facilitates work with the HP 3000 as the user does not have to wait for the HP prompt.
- Display of 132 positions on the screen with EGA boards or by horizontal rolling of up to 10,000 characters.
- In contrast to most terminal users, PC users can switch to the world of colors even with HP\3000 programs.
- Terminal emulation is programmable. Therefore, applications can be automated and the HP is relieved from CPU-intensive UDCs. For example, with Reflection's integrated command language, the PC can be forced to automatically select a modem at a particular time, to register with a HP 3000 and to start a host application.
- Reflection can be used to produce menus from which both PC programs and HP applications can be started. Therefore, the user has a uniform user interface.
- A software printspooler is also available with Reflection so that work can continue on the PC during printing on a PC printer.
- The display memory of a PC is not limited to 2 - 8 pages, it is only limited by the PC storage. Therefore, you can roll hundreds of lines forwards and backwards.

Many of these points are unknown or unfamiliar to EDP personnel.

In addition to terminal emulation, Reflection is an efficient communication programm, a PC editor, a printer spooler, a graphics tool and a PC backup program.
When Reflection is used as a communication program, data can also be transferred between two PCs (for example because of different disk formats) or data can be exchanged via a PC between different HP or DEC computers or BBSs can be accessed with Reflection.

2. File transfer

HP data can easily be further processed on a PC when corresponding programs are not available on the HP 3000. The load on the HP 3000 is shifted when evaluations are developed and carried out on PCs.
Furthermore, PCs can perform certain tasks better than the HP 3000, for example creating graphics or performing optimization calculations.

How many PCs are only used during working hours? Outside working hours, they could work for the HP 3000.
To do this, the transfer of HP data to PCs must be planned and organized.

In order that only (little) changed data must be transferred, programs should be developed or bought which extract changed data from HP data bases and transfer them automatically to PCs without any user intervention. HP programs should be adapted so that changed data are additionally written in special files for filetransfer to the PCs. They can already be formatted on the HP in such a way that they can be taken over in the PCs spreadsheets or database e.g. dBaseIII without having to be further converted.

Of course, the data cannot be transferred onto hundreds of PCs, however, in PC networks HP data can be easily mirrored onto file servers. Some efficient PCs in departments or computer centers can also be supplied with HP data. Then query jobs and reports can be shifted from HP 3000 to PCs.

Diskstorage on PCs costs less than 25% of the price of HP disk storage. Therefore, PC disks can be used for archiving HP data.
When archiving on HP tapes, the data cannot be made as easily available for evaluations as when using PCs. PC removable disks are especially suitable for this purpose (e.g. Tandon 'Data Pack' with 30 MB per component at the price of 400 US$).

An example of the time needed for data transmission:
If 10,000 data changes arise daily with 100 bytes, the contents of about a 1.2 MB disk must be transferred to PCs. With 19200 bauds, the transmission time amounts to approx. 8 minutes.

If the data are to be transferred during the day, this can take place without hindering the work of the PC user because the data can be transferred in the background. Therefore, the PC can be used, e. g., word processing in the foreground and simultaneously file transfer in the background (multitasking).
This means the HP can access the PC without the PC user having to intervene. In addition to this data transmission (with error checking), all data recorded on the display can also be logged on a PC disk and are immediately available for further processing on the PC.

Of course, the file transfer from the PC to the HP 3000 is just as easy. Thus, PC tools can be used for data entry and the captured data can be automatically retrieved from the HP 3000 in the background.

3. Improving HP applications using PC utilities

The possibilities already described do not require any PC or HP 3000 programs in addition to the terminal emulator.

Some useful PC programs which can be used constructively with HP applications are described in the following section.

The graphic in supplement 1 shows that all entries from the keyboard and all display outputs of the HP 3000 must run through the terminal emulator. It acts as a filter and certain character sequences are recognized by Reflection and forces the PC to perform certain tasks.

This graph also shows that other programs in addition to the terminal emulator can be stand by in the PCs memory to be activated by pressing a hot key within a PC or even HP application. By this means, then PC and HP processors are actually used simultaneously.

Examples:

- Reflection can be loaded a second time as an additional program (second port required). When additional software (DesqView) and ports are available, even four connections to the HP 3000 or DEC computers can be active at the same time. By pressing one key, the user can switch over from one application to another.

- The keyboard can be reconfigurated with Reflection so that at the touch of one key many characters can be keyed in. This can save a lot of time when entering data or during program editing.
  In addition, there are further keyboard enhancer such as Prokey and DesqView which automatically learn the keyboard layout (keyboard recording) or like 'Cruis Control' can speed up the cursor dramatically. This especially simplifies the writing of programs.

- On-line help systems also belong to a state-of-the-art user interface. Without programming effort, standard PC programs can be used as a help system. You only need to record the help text and the user can already call this help within HP programs by pressing a hot key.
  (Examples of help systems are: Instant Assistant, Saywhat and Flash Up Windows).

- Sometimes the data from a HP 3000 application displayed at the terminal must be recorded for other purposes. These data are usually written down on paper re-entered into a PC for other applications. When resident PC programs are used, the HP data can be transferred directly from the display to a PC storage or directly into an application which is loaded simultaneously on the PC (cut and paste).

  A tool specially suited for this purpose is the memory-resident spreadsheet LUCID 3D. However, DesqView, Sidekick, Take Over, Magic Mirror or Tornado-Notes are also suitable. This data transmission can also be computerized with keyboard macros.
  With the aid of such programs, a user can also immediately record HP error messages and can offer the programmer information which can assist debugging.

- Data from the HP can be immediately converted into business graphics (e. g. 'Graph in the box') by memory-resident graphic programs.

- Word processing can be performed much better on a PC than on the HP\3000. Better programs are also available for the editing of programs on the PCs than on the HP. Writing small (text) modules the filtransfer from PC to HP is scarcely noticed. These programs can also be integrated into HP applications in order to be called by HP applications.

- Desktop Utilities
  Some further help functions can facilitate the users work. E.g. at any time, a pocket calculator can be loaded on top of the HP-application and the results can immediately be taken over in an input field of a HP program without additional writing. Help functions are available for the user for the conversion to hexadecimal, octal or binary. A table of all ASCII characters can be called during running any program.

  With TSR ('terminate and stay resident') phone programs, the PC can dial phone numbers directly out of a HP screen (for example Sidekick, NyTalk).

DesqView 2.0 turned out as best universal tool. Several useful utilities are combined in one program: - Multitasking-Environment, -Keyboard Macros, -Cut and Paste, -DOS-Shell, -Task-Switching, -Menue-System, -Autodialing, altogether for 129.00 US$. In connection with a Memory Board (e. g. Rampage-286), Multitasking can be carried out even above the 640K DOS limit. With a DesqView addition, the 'protected mode' of 80386 processors is also supported.

4. Resource Sharing

Most PCs are equipped with a printer. These printers can be also used by the HP in order to print out lists immediately on the working place where they are needed.
If no printer is connected to the PC, the PC operator can also use the HP printer, without having to buy additional programs next to Reflection.

The printout is easily carried out through file transfer (possible in the background) on the HP printer. However, this should be avoided because of the HPs' capacity.
We have connected 5 PCs, the HP 3000 and two printers (laser and matrix printer) to a small computer (Logical Connection, price approx. 500.00 US$). Each PC and the HP 3000 can now use both printers.

If possible, the PCs should store evaluation data in addition to the HP. By this means, the HP is relieved from evaluations. This double storage inreases data security and the end users can also generate calculations or reports when the HP does not work. The additional memory costs on the PC are low (for example 300 MB disk for 2600.00 US$).

5. Distributed processing

Upon development of new HP applications, it must be kept in mind in which way the PCs can be included in this process.
Programmers should consider to perform parts of the processing on the PC and to use the HP only for central required processing.
Of course all users of this new application must get PCs.
The PC programs can have direct access to the HP with specific assembler routines when data are not available on the PC.

Meanwhile, the developers of Reflection expanded their terminal emulator also by PPL features (Program to Program Link).
These permit the access to the loaded Reflection via PC programs.
The called Reflection PPL routines keep the connection to the HP.
The PC programs can then be developed in such a way that the required data are loaded to the PCs in larger intervals. If during the daily work the required data are not found in the PC data base, the PC accesses directly the host data base.

With the aid of multiple tools which are available for PCs, programs can be developed on PCs much more quickly. Due to reasonably priced PC screen and program generators, the development of PC software is even cheaper than for HP 3000.
The creation of reports can be accomplished especially quickly on a PC.
The programming of a simple report with sort and two joined data bases can take place within two minutes.

With a powerful PC (HP Vectra RS 20), printing a report of 5000 records (into a spooler) with sort takes less than three minutes. Including file transfer (if the data are not yet on the PC), the first creation of this report takes approximately 10 minutes. (Program: Relational Report Writer).

A fool proof program for updating master files with 20 fields, including check against two other data bases, generic key search on two files and a few testings can be finished with the aid of a PC program generator in less than 30 minutes. (Programming language and data base: Foxbase+, own program generator).

There are a few programming systems which can be already used on PC and on the HP 3000 and which therefore allow applications running on PC and HP 3000. Since this tools are very expensive, the use of lowcost PC tools is more economical in many cases. There are reasonably priced PC products which can already be used on different system. The above mentioned Foxbase+ runs under MS DOS, Unix (SCO-Xenix), in the 'protected mode' on 80386 processors.

Distributed data processing can also mean storage of HP screens on the PC. To do this, Reflection offers 'Formscaching', as on the HP terminal 2624. Furthermore screens can be taken over from other PC screen generators. This screens can be activated by short commands from the HP.
The transfer of screens can be avoided while working with remote computers. Thus the application runs more quickly on a PC an cuts data transmission costs.

All the examples described can be implemented immediately without great financial effort.

The control of the PC can always be kept by the HP. Because of the Reflection commands, the HP can monitor the PC. The only requirement is, that the PC is switched on and Reflection is loaded. An example in which way PC commands are issued by the HP is shown in supplement 4.

More and more standard HP programs are able to access PC features in conjunction with Reflection commands(e. g. 'Qedit' by Robelle, 'Hello 3000' by Sydes).

By suitable own programs, you can take the possibility to link to PCs which are just in a session and even to those which are not logged in.
With the aid of 'Tell' and 'Warn' commands of the MPE, one cannot transmit commands via Reflection to the PCs, since the MPEs tell/warn allows no escape sequences. Therefore, an own program must be written or the MPE must be patched. (The patching of the Tell/Warn commands was exactly described in a Interact magazine).

6. End user support during planning and creation of applications

Meanwhile more and more users are acquainted with PC programs like dBaseIII, Lotus or Symphony. Thus they can solve many problems by themselves when the data from the HP are prepared for them.
In order to avoid an unorganized and not documented second EDP, the EDP department should guide the users and train them in the application.
Suitable aids for the documentation and the data backup must be specified and its use should be monitored.

Special PC training programs can effectively support this training. These programs, for example 'Instant Replay' by Nostradamus or 'Dan Bricklin's Demo-Program' can 'take photos' from existing applications and combine them for interactive demos.

New applications can be generated and tested as prototypes already before creation on the PC together with the users. Thus, the appliccation is exactly decribed.

7. Problems with integration

In order to avoid problems, the use of PCs must be well prepared. Supplement 2 shows a checklist of important subjects.

Data security is especially important when using PCs. It must be ensured by monitor programs that the users cannot install own programs. Viruses on the PC may be the result of unexamined software.
In addition, hidden programs could be used in order to record the number of key strokes (keyboard recorder such as 'Instant Replay'). Thus all passwords can be found out and the data security would be endangered.

In order to minimize the training effort, the user interfaces of PCs and hosts should be uniform.
Every PC application can be started with the aid of Reflection out of an HP program. Thus an uniform user interface could be installed on the host (HELLO 3000 has this interface).
An HP system offers the advantage to be used also by normal terminals.
In order to relieve users who are already acquainted with the HP from learning the MS-DOS commands, I have programmed a small command interpreter in Turbopascal for PCs, which offers a similar user interface like MPE on the HP, for example 'LISTF . ' instead of 'DIR *.*' or 'PURGEACCOUNT' instead of 'REMOVE DIRECTORY', with 'REDO', 'NEWACCOUNT', passwords and so on.

Upon redevelopment of HP programs, one should learn from existing PC programs and adapt the user interface to well-known PC programs.

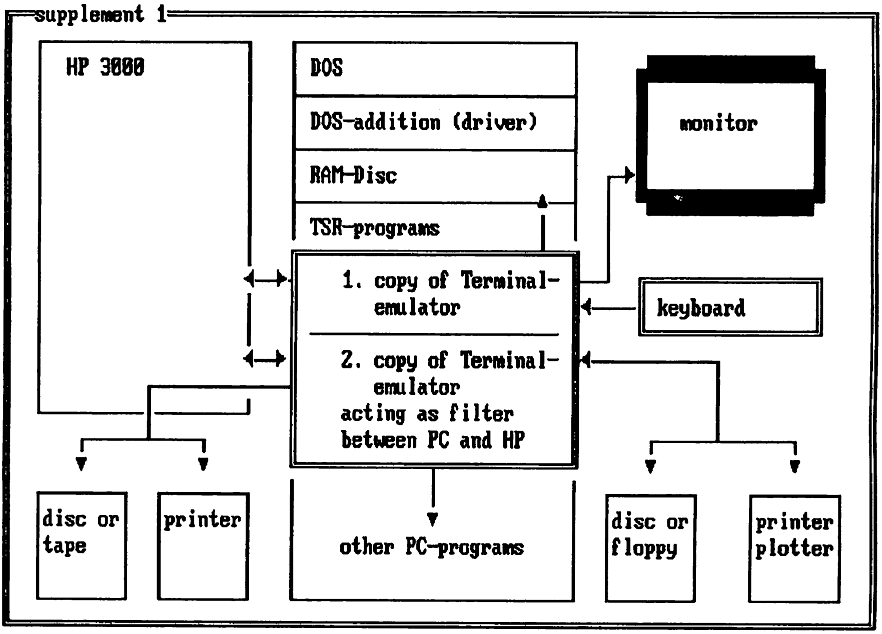
If you use many PCs, you should take into consideration that the update of the PCs is carried out from the host. Using hundreds of PCs can no longer mean wandering around with the floppy from one PC to the next in order to install a new release.
An HP monitor should check regularly whether unallowed programs are available on a PC and whether the disk capacity of a PC is still sufficient. In addition, the HP can transfer a program to the PCs via file transfer which collects all required data and transfers them to the host. Here these data can be checked.

In addition, a PC support program must be available for offering remote support for PCs (for example Carbon Copy).

Enclosures

1. Diagram HP/PC/Terminal Emulator Environment
2. PC integration checklist
3. Why PCs should be used instead of dumb terminals
4. Example Cobol program for accessing PCs by HP

Supplement 1 (page 1 of 1); System Overview PC/HP/Terminal emulator


supplement 1
HP 3000
DOS
DOS-addition (driver)
RAM-Disc
TSR-programs
monitor
1. copy of Terminal-emulator
2. copy of Terminal-emulator acting as filter between PC and HP
keyboard
disc or tape
printer
other PC-programs
disc or floppy
printer plotter

c:\instant\doc\bild1 Rolf Schleicher 19.06.88

Supplement 2 (page 1 of 2): Checklist for PC Integration

1. There should be a configuration file on the HP for each terminal or PC. This file should contain information about the hard and software used at each desk (workplace) so that all HP programs have information available about which resources can be used, for example: slave printers, PC programs, disk space etc. On the PC there should also be a configuration file wich can be updated by a PC program and transferred to the HP3000.
2. Begin by developing the procedure to update the PCs by HP.
3. If you plan downloading data to the PC and uploading after changing to HP, you must have a logging strategy. Data entries on HP should have additional fields which show whether the downloading to the PC was successful (date/time, etc.).
4. There must be backup-PCs available with all disk space free.
5. Train some (interested!) people as PC specialists.
6. Reserve a budget for buying PC programs and for testing their features.
7. At each PC an 'SOS' start disk must be available for the case when the hard disk doesn't work, so that at least the terminal-emulation can be carried out.
8. Train the end user to use the PC and its programs efficiently.
9. Keep in contact with other PC users. Share your knowledge.
10. Supervising of PC disk space is necessary. New programs need a lot of space and the PC disks are soon full of garbage.
11. Each PC needs a lot of floppies for a backup, at least 100 floppies for a 20 MB-disk (two generations of backups on 360KB-floppies).
12. Coordinate your printers (with switches like LOGICAL-CONNECTION).
13. PCs should have colour screens and graphic capability.
14. Design your workplace carefully (PC cooling fans are noisy and must not blow into colleagues' faces).
15. Plan PC maintenance strategy (backup PCs are often more economical than maintenance-contracts).
16. Control the software licenses centrally in order to keep track of software updates. Keep the original floppies of installed software in a central place.
17. Pay attention to data security (passwords, security system).

18. Buy cheap (laptop) PCs for training end users (and for taking home at weekends).
19. Subscribe to PC magazines.
20. All your programmers should work with (AT-compatible) PCs with colour screens.
21. Develop your own integration and software strategy.
22. Work out software programs which are transportable to other computers.
23. The following should be organized and controlled: user interfaces, help systems, use of colours, meaning of function keys and screen design.
24. Plan your end user support.
25. Always consider whether PCs could be used instead of HP (HP as file server).
26. Share the end users' experiences by holding user meetings and creating a PC newspaper.
27. Build up an extensive library of books and software.
28. Try to use the same commands and user interface design on both the PC and HP.
29. Consider using removable hard disks or floppies with at least 10 MB capacity (Tandon PAC, Bernoulli-drives).
30. Backup the directory structure as well as the data. The former is more important!
31. Do not carry out software developments without using PC prototyping and creating demos and tutorials.
32. Train the users to avoid possible pitfalls of end user tools.

Supplement 3 (page 1 of 2): Why use PCs instead of dumb terminals

---

Some arguments for changing from the 'buy terminal' to 'buy PC' strategy.

1. Terminals often only work with a special computer. If you change the computer brand (this happens sometimes), you may have terminals which don't work with the new computer.
2. If you have two computers for example HP and DEC or HP and IBM, you need two terminals on your desk. A PC can be connected to more than one computer at the same time with hardware and/or software.
3. PCs offer the user a wider range of possibilities than terminals. This is the reason why they should be used even when a PC itself is not necessary for current applications. Here is a list of features you have available at a PC terminal with REFLECTION terminal emulation:

3.1 Formscaching: screens can be stored in the PC memory and must not be transferred between HP and PC. This saves time and money used for remote computing and is highly recommended for taking power from the HP.

3.2 Some resource-consuming procedures (for example UDCs) can be performed by the PC.

3.3 You can switch on the 'type-ahead' feature of the PC terminal in order to type as fast as you can without having to wait for the HP's prompt.

3.4 REFLECTIONS built-in printspooler allows you to continue working while the slaveprinter is printing.

3.5 Horizontal scrolling allows you to see wide lines on the PC. With high-resolution screens you can display 132 columns without scrolling.

3.6 You can use a pointing device (for example mouse) to move the pointer on the screen and to carry out vertical and horizontal scrolling.

3.7 You can keep hundreds of lines in the PC's memory which can be scrolled back and forth (also with the HP150!)

3.8 For little money, you can have colour and graphics.

3.9 Data from the screen can be directly transferred to the PC storage, so that error messages and data must not be written down and can be directly computed with PC programs.

3.10 The keyboard can be designed by the user or extended by the HP-program to support all kinds of applications.

3.11 During HP applications, at the touch of a key, the user always has another computer available which can perform certain tasks much better than HP programs (for example word processing, graphics, spreadsheets).

3.12 The PC can be controlled by other computers (like HP3000) and can be used at night as a co-processor for creating graphs or calculating inear programming.

3.13 With a PC terminal you can easily switch over to more than one application without having to finish one job before starting a new one. If you have four serial ports on your PC, you can switch between four HP-applications and, if necessary, additional PC programs.

3.14 Cheap printers, connected to the PC, can print out at your desk just the lines you need. The printers can also be shared with the HP.

3.15 Data sharing between HP and PC is easy when automatic background file transfer is used.

3.16 The PC can control the HP. With the simple REFLCTION command language, the PC is able to send commands to the HP at pre-defined times.

3.18 If IBM computers are connected to HP via SNA/IMF, IBM users can use all these features without expensive coax cabelling or interface cards in the PCs.

I thank my wife for translating my lecture with the computer translation system "LOGOS", installed on a WANG VS/85.

All mistakes in this text are made by the WANG-computer.

Supplement 4

```
$CONTROL USLINIT

*this is an example of a cobol-program which receives input via
*the info-parameter of the run-command and sends this
*command with added prefix back to then terminal-emulator.
*Author: Rolf R. Schleicher, Harksheider Str. 29, D2000 Hamburg 65

 IDENTIFICATION DIVISION.
 PROGRAM-ID. PC.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.

 01  PC-VARIABLE PIC X(80).
 01  REFLECTION.
     05  ESCAPE            PIC X VALUE %33.
     05  REFLECT-SEQUENCE  PIC X(3) VALUE "&oC".
 01  INFO-LENGTH        PIC S9(4) COMP VALUE 80.

 PROCEDURE DIVISION.
 START-p1 SECTION .
     MOVE SPACES TO PC-VARIABLE.
     call intrinsic "GETINFO" using pc-variable
                             INFO-LENGTH \\.
     IF INFO-LENGTH = 0
        DISPLAY ESCAPE "H" ESCAPE "J"
        DISPLAY
        "CORRECT SYNTAX: RUN PC;INFO=" QUOTE "command" QUOTE
        DISPLAY "-------------------------------------------"
        DISPLAY "This program takes your input from the"
        DISPLAY "INFO-parameter of the RUN-command."
        DISPLAY "It adds the REFLECTION-prefix '<esc>&oC' to"
        DISPLAY "your command an sends it back to your PC."
        DISPLAY "-------------------------------------------"
        DISPLAY "With RUN PC;INFO=" QUOTE "HELP" QUOTE " you"
        DISPLAY "will get a list of REFLECTION-commands"
        DISPLAY "-------------------------------------------"
        DISPLAY " ".
*--------------- calling REFLECTION ------------------
*switching off standard terminal response (S,U or F)
     DISPLAY REFLECTION "SET DISABLE-COMP-CODES YES".
*----------------sending your command via REFLECTION ---
     DISPLAY REFLECTION PC-VARIABLE.
     STOP RUN.

*Example for using the program:
*------------------------------
*SYNTAX:   RUN PC;INFO="your command"
*1. the MPE-command      :RUN PC;INFO="DIR *.*"
*   causes the PC to show the PC-directory on the HP-screen.
*2. the MPE-command      :RUN PC;INFO="SHELL FORMAT A:"
*   turns the PC to DOS, starts formatting a floppy in driv A:
```